

ENKCB-1.0

VAX 11/730	CPU
MICRO PART	2

EP - ENKCB - DL - 1.0
FICHE 1 OF 2

MAY 1982
COPYRIGHT © 1982
MADE IN USA

digital

ENKCB-1.0

VAX 11/730 CPU
MICRO PART 2

EP-ENKCB-DL-1.0
FICHE 2 OF 2

MAY 1982
COPYRIGHT © 1982
MADE IN USA

disi21

IDENTIFICATION

Product code: ZZ-ENKCB VERSION 1.0
Product name: MICRO-DIAGNOSTIC FOR VAX-11/730 CPU MODULE
Product date: 8-MARCH-1982
Maintainer: BASE SYSTEMS DIAGNOSTIC ENGINEERING

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Copyright (c) 1982 by Digital Equipment Corporation. All Rights Reserved.

The following are trademarks of Digital Equipment Corporation.

DEC
DECUS
UNIBUS

DECsystem-10
MASSBUS
VAX

DECSYSTEM-20
PDP
VMS

d i g i t a l

Table of Contents

1.0	ABSTRACT	3
2.0	HARDWARE REQUIREMENTS	3
3.0	SOFTWARE REQUIREMENTS	3
4.0	PREREQUISITES	3
5.0	OPERATING INSTRUCTIONS	3
5.1	Options	3
5.2	Event Flags	4
5.3	APT Setup	4
6.0	PROGRAM FUNCTIONAL DESCRIPTION	4
6.1	Program Overview	4
6.2	Program Size	4
6.3	Program Run Times	4
6.4	Fault Detection	4
6.5	Performance During Hardware Failures	5
6.6	Program Applications	5
6.7	Test Descriptions	5
7.0	MAINTENANCE HISTORY	6

1.0 ABSTRACT

This program is a micro-diagnostic designed to test the hardware on the CPU module of the VAX-11/730 processor.

2.0 HARDWARE REQUIREMENTS

This program will run with the minimum hardware configuration of a WCS, and a CPU.

Other options (and an MCT and Array Module) may be installed without affecting the diagnostic's performance.

3.0 SOFTWARE REQUIREMENTS

This diagnostic must be run under the VAX-11/730 micro monitor (ENKAA) in a standalone environment. The micro-diagnostic is written into WCS RAM, wiping out any system micro-code that may have been resident. The micro-diagnostic monitor takes up the area where the console software is normally present.

4.0 PREREQUISITES

The WCS module and a portion of the CPU are assumed to have been tested and known to be good before this diagnostic is run. The programs ENKBA through ENKBE, and ENKCA will perform this testing.

5.0 OPERATING INSTRUCTIONS

This program may be executed by typing DI SE ENKCB after the micro-monitor has been loaded and the MIC> prompt has been printed at the terminal. Both the micro-monitor and this program are loaded from a TU58 cassette or downline loaded through the APT-RD port. See the OVERVIEW MANUAL for more information.

5.1 Options

Not applicable

5.2 Event Flags

Not applicable

5.3 APT Setup

The following describes the APT parameters needed to execute this diagnostic:
TBS

6.0 PROGRAM FUNCTIONAL DESCRIPTION

6.1 Program Overview

This program is designed to detect stuck-at faults and provide a repair tool which helps isolate the failing component. A bottom up diagnostic strategy is utilized which builds on previous tests. The tests should be run in consecutive order, to gain the best isolation of a failing component.

6.2 Program Size

This program runs in WCS RAM memory and is approximately 48K bytes long (utilizing about 16000 micro-words in WCS).

6.3 Program Run Times

This program will take approximately 9 seconds to run including initialization time and exclusive of load time.

6.4 Fault Detection

The diagnostic will report errors with the following header:
SECT TST ERR EXP REC OTHER MSK MODULE

1. SECT is the program name (ENKCB)
2. TST is the test number that failed.
3. ERR is the error number that failed.

4. EXP is the expected (correct) data.
5. REC is the received (actual) data.
6. OTHER is any other pertinent data (such as an address). This field is not always used. N/A will be printed under OTHER when not in use. The ERRORS section in the listing will describe the contents of this field when it is used.
7. MASK is the error mask used to check the result. Bits set to a 1 in this mask correspond to bits in the result that are not checked.
8. MODULE is the suspected module that caused the failure.

6.5 Performance During Hardware Failures

A power fail will cause a rebooting of the system. Other failures, such as a timeout or a WCS parity error, will print an error message and return to the MIC> prompt. The operator can then issue other commands, including a continue if desired.

6.6 Program Applications

This program can be used by field service to determine which module should be replaced and to verify that the replacement eliminated the failure. It can also be used as a repair tool to help isolate a failing component by manufacturing, field service or engineering. The program is APT compatible.

Customers may use the program to verify the basic integrity of the central processor module in the system.

6.7 Test Descriptions

See the actual test in the listing for detailed descriptions and error analyses. A brief description of the logic being tested by each test can be found in the table of contents of the listing.

ZZ-ENKCB-1.0 Documentation
ENKCB MICRO-DIAGNOSTIC FOR VAX-11/730 CPU MODULE
MAINTENANCE HISTORY

G 1
Page 6
01 Mar 82

Fiche 1 Frame G1

Sequence 6

7.0 MAINTENANCE HISTORY

DATE	VERSION	DESCRIPTION
8-MAR-82	1.0	Initial release.

46	FIELD DEFINITIONS FOR MAIN MICRO WORD	VER 00.08
1052	MACRO DEFINITIONS	VER 00.02
1708	FIELD DEFINITIONS FOR DATA PATH CONTROL MICRO WORD	VER 00.01
1761	CONTROL ROM CONTENTS	VER 00.01
2510	DIAGNOSTIC MACROS AND DEFINITIONS	
3167	COMMON LS ASSIGNMENTS (TO REGISTERS)	
3193	PROGRAM SPECIFIC LS ASSIGNMENTS (LS 80-F7)	
3222	Microinstruction Formats	
3605	Tables	
3736	2901 ALU Control Description	
3863	TEST POINTERS	
4017	DIAGNOSTIC POINTERS	
4106	LS DATA SECTION	
4497	PROGRAM SPECIFIC DATA SECTION (LS 80-F7)	
4876	WCS SUBROUTINES FOR CONSOLE USE	
4877	GENERATE TRANSFER DATA	
4921	DEPOSIT MCT CSR ROUTINE	
4987	EXAMINE MCT CSR ROUTINE	
5074	DEPOSIT MCT TRANSLATION BUFFER ROUTINE	
5190	EXAMINE TRANSLATION BUFFER ROUTINE	
5253	DEPOSIT UNIBUS MAP ROUTINE	
5351	EXAMINE UNIBUS MAP ROUTINE	
5414	DEPOSIT MAIN MEMORY ROUTINE	
5472	EXAMINE MAIN MEMORY ROUTINE	
5534	EXAMINE IDC ROUTINES	
5600	DEPOSIT IDC ROUTINES	
5672	SAVE WR ROUTINE	
5723	RESTORE WR ROUTINE	
5774	WCS SUBROUTINES FOR CPU USE	
5775	ERROR ROUTINE	
5942	START TRANSFER ROUTINE	
6032	General Use Subroutines	
6080	TEST 1 - Stack RAM Data Test (M8390 CPU Module)	
6360	TEST 2 - Nested Subroutines Test (M8390 CPU Module)	
6524	TEST 3 - Loop control (jump to stack) Tests (M8390 CPU Module)	
6728	TEST 4 - OS 'ORed' with NAD 0-4 (M8390 CPU Module)	
6877	TEST 5 - CONS HALT, Skip and Jump on Interrupt (M8390 CPU Module)	
7071	TEST 6 - INTR CTLR PAL, IPL, and BUS IRQ Test (M8390 CPU Module)	
7546	TEST 7 - PRI ENC Chip Test (M8390 CPU Module)	
7733	TEST 8 - T-TRAP and Mask Test (M8390 CPU Module)	
8100	TEST 9 - SHIFT DATA PAL SI=ASH.WR Test (M8390 CPU Module)	
8196	TEST A - SHIFT DATA PAL SI=ASH.WR.Q Test (M8390 CPU Module)	
8415	TEST B - SHIFT DATA PAL SI=SHF.WR.Q Test (M8390 CPU Module)	
8515	TEST C - Signed Multiply (SI=MUL) Test (M8390 CPU Module)	
8817	TEST D - Unsigned Multiply (SI=UMUL) Test (M8390 CPU Module)	
8950	TEST E - Sign Extend Logic Test (M8390 CPU Module)	
9033	TEST F - DEST CTL ROM And SRC.ALU CTL PAL Tests (M8390 CPU Module)	

```

:1  TITLE "ENKCB Micro-diagnostic for DAP module Rev 01.00"
:2
:3
:4  COPYRIGHT (c) 1982 BY
:5  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
:6  MASSACHUSETTS. ALL RIGHTS RESERVED.
:7
:8  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
:9  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE INCLUSION
:10 OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER COPIES THEREOF
:11 MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON. NO
:12 TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY TRANSFERRED.
:13
:14 THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE, AND
:15 SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
:16
:17 DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
:18 SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.
:19
:20
:21 ++
:22 FACILITY: VAX-11/730 MICRO-DIAGNOSTIC.
:23
:24 ABSTRACT: This micro-diagnostic tests the hardware of the DAP module
:25 in the VAX-11/730 processor.
:26
:27 ENVIRONMENT: ENKAA Micro-diagnostic Monitor
:28
:29 AUTHOR: Mike Bibeault 4-MAY-81
:30
:31 MODIFIED BY: David Mayo 8-MAR-82 VERSION 01.00
:32 Initial release.
:33
:34 --
:35 .nolist
:39 .list
:40 .NOCREF
  
```

```

:41 .NOBIN
:42 .SEQUENTIAL
:43 .RTOL
:44 .HEXADECIMAL
  
```

```

:45 .PAGE
:46 .TOC "FIELD DEFINITIONS FOR MAIN MICRO WORD VER 00.08"
:47 .SET/UPC=<000>
:48
:49 ; THE FOLLOWING EXPRESSIONS ARE VALIDITY CHECKS FOR FIELD COMBINATIONS
:50
:51 .SET/A.VAL=<.CASE[<OPC2/>]OF[0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0]>
:52 .SET/B.VAL=<.CASE[<OPC2/>]OF[0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1]>
:53 .SET/D.VAL=<.CASE[<OPC2/>]OF[0,0,0,1,0,0,0,0,1,1,1,1,1,1,1,1]>
:54 .SET/XD.VAL=<.EQL[<OPC1/>,<OPC1/MOVE>]>
:55 .SET/CC.VAL=<.CASE[<OPC2/>]OF[0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1]>
:56 .SET/DT.VAL=<.EQL[<OPC2/>,<OPC2/MEM.REQ>]>
:57 .SET/SCTL.VAL=<.CASE[<OPC2/>]OF[0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1]>
:58 .SET/JCTL.VAL=<.CASE[<OPC2/>]OF[1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0]>
:59 .SET/JUMP.VAL=<.EQL[<OPC2/>,<OPC2/JUMP>]>
:60 .SET/DECODE.VAL=<.EQL[<OPC2/>,<OPC2/DECODE>]>
:61 .SET/MISC.VAL=<.EQL[<OPC2/>,<OPC2/MISC>]>
:62 .SET/DP.VAL=<.EQL[<OPC/>,<OPC/BASIC>]>
:63
:64 ; THE FOLLOWING VALIDITY CHECKS ARE USE FOR THE PAGE BOUNDARY PROBLEM.
:65
:66 .SET/PAGE.PROB=<.EQL[<.AND[<7FF>,<.]>,<7FE>]>
:67 .SET/SKIP.LEGAL2=<.OR[<.EQL[<SCTL/>,<SCTL/RET.NOERR>]>,<.EQL[<SCTL/>,<SCTL/POP.USTACK>]>]>
:68
:69 .SET/SKIP.LEGAL=<.OR[<.EQL[<SCTL/>,<SCTL/NO.SKIP>]>,<.EQL[<SCTL/>,<SCTL/RETURN>]>,<.EQL[<SCTL/>,<SCTL/RETURN+1>]>,<SKIP.LEGAL2>]>
:70
:71 .SET/SKIP.PAGE.VAL=<.CASE[<PAGE.PROB>]OF[1,<SKIP.LEGAL>]>
:72 .SET/JUMP.CHECK=<.OR[<.EQL[<JCTL/>,<JCTL/JSR>]>,<.EQL[<JCTL/>,<JCTL/JSR.VALID>]>]>
:73
:74 .SET/JUMP.PAGE.VAL=<.CASE[<PAGE.PROB>]OF[1,<.XOR[1,<JUMP.CHECK>]>]>
:75
:76
:77 ; MICRO INSTRUCTION OPCODE DEFINITIONS
:78
:79 OPC/=<22>,.DEFAULT=<OPC/BASIC> ; MICRO OPCODE FIELD FOR SINGLE BIT OPCODE.
:80
:81 BASIC=1 ; OPCODE FOR 'BASIC' MICRO INSTRUCTION
:82
:83 OPC1/=<22:20> ; MICRO OPCODE FIELD FOR THREE BIT OPCODES.
:84
:85 EXTENDED=2 ; OPCODE FOR 'EXTENDED' MICRO INSTRUCTION
:86 MOVE=3 ; OPCODE FOR 'MOVE' MICRO INSTRUCTION
:87
:88 OPC2/=<22:19> ; MICRO OPCODE FIELD FOR FOUR BIT OPCODES
:89
:90 MEM.REQ=3 ; OPCODE FOR 'MEM.REQ' MICRO INSTRUCTION
:91 MISC=2 ; OPCODE FOR 'MISC' MICRO INSTRUCTION
:92 JUMP=1 ; OPCODE FOR 'JUMP' MICRO INSTRUCTION
:93 DECODE=0 ; OPCODE FOR 'DECODE' MICRO INSTRUCTION
:94
:95
:96 ; THE FOLLOWING FOUR FIELDS ARE RELATED TO ADDRESSING
:97 ; VARIOUS RAM MEMORIES WITHIN THE DATA PATH.
:98
:99 ; THE A.ADRS REFERS TO THE 2901 INTERNAL RAM 'A-PORT'

```



```

:100 : THE B.ADRS REFERS TO THE 2901 INTERNAL RAM 'B-PORT'
:101 : THE XB.ADRS REFERS TO THE 2901 INTERNAL RAM 'B-PORT'
:102 : THE D.ADRS REFERS TO THE LOWER 128 LOCATIONS OF THE LOCAL STORE RAM
:103 : THE XD.ADRS REFERS TO ALL 256 LOCATIONS OF THE LOCAL STORE RAM
:104 :
:105 : THE VALIDITY STATEMENTS ALLOW USAGE OF THESE FIELDS IN THE FOLLOWING MANNER
:106 :
:107 :     A.ADRS          EXTENDED MICRO INSTRUCTION
:108 :
:109 :     B.ADRS          BASIC MICRO INSTRUCTION
:110 :                     EXTENDED MICRO INSTRUCTION
:111 :                     MOVE MICRO INSTRUCTION
:112 :                     DECODE.IS MICRO INSTRUCTION
:113 :
:114 :     XB.ADRS          MOVE XWR MICRO INSTRUCTION
:115 :
:116 :     D.ADRS          BASIC MICRO INSTRUCTION
:117 :                     MEM.REQ MICRO INSTRUCTION
:118 :
:119 :     XD.ADRS          MOVE MICRO INSTRUCTION
:120 :
:121 : A.ADRS/=<10:9>,.VALIDITY=<A.VAL>
:122 :
:123 :     MASK=3          ; REGISTER BACKUP MASK
:124 :
:125 : B.ADRS/=<8:7>,.VALIDITY=<B.VAL>,.DEFAULT=0
:126 :
:127 :     MASK=3          ; REGISTER BACKUP MASK
:128 :
:129 : ; THIS ADDRESS FIELD WILL CONTAIN THE LOW TWO BITS OF THE 2901 B ADDRESS
:130 : ; THE THIRD BIT WILL BE FORCED BY HARDWARE.
:131 :
:132 : XB.ADRS/=<8:7>,.VALIDITY=<.EQL[<OPC1/>,<OPC1/MOVE>]>
:133 :
:134 :     BPC=0           ; ADDRESS OF BEGINNING PC      WR[4]
:135 :     VA=1            ; ADDRESS OF MEMORY REFERENCE WR[5]
:136 :     VAR=1           ; ADDRESS OF LAST MEMORY REFERENCE. (WR[5] IN 2901).
:137 :
:138 : ;D.ADRS/=<15:9>,.VALIDITY=<D.VAL>,.DEFAULT=0
:139 :
:140 :
:141 :     SAVED.2ND.REQUEST=0 ; MM SAVED STATE FOR REQUEST.
:142 :     T1=1              ; TEMP LOCATION 1
:143 :     T2=2              ; TEMP LOCATION 2
:144 :     T3=3              ; TEMP LOCATION 3
:145 :     T4=4              ; TEMP LOCATION 4
:146 :     T5=5              ; TEMP LOCATION 5
:147 :     T6=6              ; TEMP LOCATION 6
:148 :     T7=7              ; TEMP LOCATION 7
:149 :     T8=8              ; TEMP LOCATION 8
:150 :     T9=9              ; TEMP LOCATION 9
:151 :     T10=0A            ; TEMP LOCATION 10
:152 :     BACKUP.CM.PC=0A    ; COMPATIBILITY MODE KEEPS ORIGINAL PC HERE.
:153 :
:154 :     T11=0B           ; TEMP LOCATION 11
    
```

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) FIELD DEFINITIONS F 3-MAR-82
FIELD DEFINITIONS FOR MAIN MICRO WORD

L 1
Fiche 1 Frame L1
ENKCB Micro-diagnostic for DAP module
VER 00.08

Sequence 11
Rev 01.00

Page 5

```
:155 : T12=0C : TEMP LOCATION 12
:156 : T13=0D : TEMP LOCATION 13
:157 : T14=0E : TEMP LOCATION 14
:158 : T15=0F : TEMP LOCATION 15
:159 : PC=10 : MACRO PROGRAM COUNTER
:160 : WR.BACKUP=11 : BACKUP WORKING REGISTER FOR MOV MEM.DATA TO WR[]
:161 : SAVED.VAR=12 : MM SAVED STATE FOR VAR
:162 : SAVED.DATA=13 : MM SAVED STATE FOR DATA
:163 : SAVED.PTE.ADDRESS=14 : MM SAVED STATE FOR PAGE TABLE ENTRIES ADDRESS
:164 : SAVED.PTE=15 : MM SAVED STATE FOR PAGE TABLE ENTRIES
:165 : T16=16 : TEMP LOCATION 16
:166 : T17=17 : TEMP LOCATION 17
:167 : IE.TEMP4=18 : INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION FOUR.
:168 : #FFFFFF00=19 : CONSTANT
:169 : #FFFFFF00=1A : CONSTANT
:170 : #FFFFFFFC=1B : CONSTANT
:171 : #FFFFFFF=1C : CONSTANT
:172 :
:173 : THIS LOCATION CONTAINS THE SOFT COPY OF THE PSL. WHENEVER THE
:174 : HARDWARE VERSION OF THE PSL IS CHANGED (PSL.HW), THEN THIS
:175 : CHANGE IS ALSO REPORTED HERE BY THE MICROCODE. ALL BITS OF
:176 : THE PSL ARE REALLY LIKE THOSE IN THIS WORD EXCEPT FOR THE
:177 : CONDITION CODES WHICH MUST BE OBTAINED FROM PSL.CC
:178 :
:179 : PSL=1D : SOFT COPY OF VAX PSL
:180 : VAR=1E : VIRTUAL ADDRESS REGISTER
:181 : VAR2=1F : VIRTUAL ADDRESS REGISTER
:182 :
:183 :
:184 : #1=20 : MASK 00000001 (HEX)
:185 : #2=21 : MASK 00000002
:186 : #4=22 : MASK 00000004
:187 : #8=23 : MASK 00000008
:188 : #10=24 : MASK 00000010
:189 : #20=25 : MASK 00000020
:190 : #40=26 : MASK 00000040
:191 : #80=27 : MASK 00000080
:192 : #100=28 : MASK 00000100
:193 : #200=29 : MASK 00000200
:194 : #400=2A : MASK 00000400
:195 : #800=2B : MASK 00000800
:196 : #1000=2C : MASK 00001000
:197 : #2000=2D : MASK 00002000
:198 : #4000=2E : MASK 00004000
:199 : #8000=2F : MASK 00008000
:200 : #10000=30 : MASK 00010000
:201 : #20000=31 : MASK 00020000
:202 : #40000=32 : MASK 00040000
:203 : #80000=33 : MASK 00080000
:204 : #100000=34 : MASK 00100000
:205 : #200000=35 : MASK 00200000
:206 : #400000=36 : MASK 00400000
:207 : #800000=37 : MASK 00800000
:208 : #1000000=38 : MASK 01000000
:209 : #2000000=39 : MASK 02000000
```

:210	:	#40000000=3A	:	MASK 04000000
:211	:	#80000000=3B	:	MASK 08000000
:212	:	#100000000=3C	:	MASK 10000000
:213	:	#200000000=3D	:	MASK 20000000
:214	:	#400000000=3E	:	MASK 40000000
:215	:	#800000000=3F	:	MASK 80000000
:216	:		:	
:217	:	BIT0=20	:	MASK 00000001
:218	:	BIT1=21	:	MASK 00000002
:219	:	BIT2=22	:	MASK 00000004
:220	:	BIT3=23	:	MASK 00000008
:221	:	BIT4=24	:	MASK 00000010
:222	:	BIT5=25	:	MASK 00000020
:223	:	BIT6=26	:	MASK 00000040
:224	:	BIT7=27	:	MASK 00000080
:225	:	BIT8=28	:	MASK 00000100
:226	:	BIT9=29	:	MASK 00000200
:227	:	BIT10=2A	:	MASK 00000400
:228	:	BIT11=2B	:	MASK 00000800
:229	:	BIT12=2C	:	MASK 00001000
:230	:	BIT13=2D	:	MASK 00002000
:231	:	BIT14=2E	:	MASK 00004000
:232	:	BIT15=2F	:	MASK 00008000
:233	:	BIT16=30	:	MASK 00010000
:234	:	BIT17=31	:	MASK 00020000
:235	:	BIT18=32	:	MASK 00040000
:236	:	BIT19=33	:	MASK 00080000
:237	:	BIT20=34	:	MASK 00100000
:238	:	BIT21=35	:	MASK 00200000
:239	:	BIT22=36	:	MASK 00400000
:240	:	BIT23=37	:	MASK 00800000
:241	:	BIT24=38	:	MASK 01000000
:242	:	BIT25=39	:	MASK 02000000
:243	:	BIT26=3A	:	MASK 04000000
:244	:	BIT27=3B	:	MASK 08000000
:245	:	BIT28=3C	:	MASK 10000000
:246	:	BIT29=3D	:	MASK 20000000
:247	:	BIT30=3E	:	MASK 40000000
:248	:	BIT31=3F	:	MASK 80000000
:249	:		:	
:250	:	GPR0=40	:	GPR 0
:251	:	GPR1=41	:	GPR 1
:252	:	GPR2=42	:	GPR 2
:253	:	GPR3=43	:	GPR 3
:254	:	GPR4=44	:	GPR 4
:255	:	GPR5=45	:	GPR 5
:256	:	GPR6=46	:	GPR 6
:257	:	CM.SP=46	:	IN COMPATIBILITY THIS IS THE STACK POINTER.
:258	:	GPR7=47	:	GPR 7
:259	:	CM.PC=47	:	IN COMPATIBILITY MODE THIS IS THE PC.
:260	:	GPR8=48	:	GPR 8
:261	:	GPR9=49	:	GPR 9
:262	:	GPR10=4A	:	GPR 10
:263	:	GPR11=4B	:	GPR 11
:264	:	GPR12=4C	:	GPR 12

```

:265 : AP=4C : ARGUMENT POINTER
:266 : GPR13=4D : GPR 13
:267 : FP=4D : FRAME POINTER
:268 : SP=4E : GPR 14
:269 : TO=4F : TEMP LOCATION 0
:270 :
:271 : ZERO=50 : ZERO CONSTANT
:272 : #3=51 : CONSTANT
:273 : #5=52 : CONSTANT
:274 : #6=53 : CONSTANT
:275 : #7=54 : CONSTANT
:276 : #9=55 : CONSTANT
:277 : #8=56 : CONSTANT
:278 : #C=57 : CONSTANT
:279 : #D=58 : CONSTANT
:280 : #E=59 : CONSTANT
:281 : #F=5A : CONSTANT
:282 : #13=5B : CONSTANT
:283 : #1F=5C : CONSTANT
:284 : #34=5D : CONSTANT
:285 : #3B=5E : CONSTANT
:286 : #3F=5F : CONSTANT
:287 : #4B=60 : CONSTANT
:288 : #66=61 : CONSTANT
:289 : #A0=62 : CONSTANT
:290 : #F0=63 : CONSTANT
:291 : #FF=64 : CONSTANT
:292 : #1FF=65 : CONSTANT
:293 : #FFF=66 : CONSTANT
:294 : #2001=67 : CONSTANT
:295 : #300C=68 : CONSTANT
:296 : #7F80=69 : CONSTANT
:297 : #C000=6A : CONSTANT
:298 : #FFE0=6B : CONSTANT
:299 : #FFFF=6C : CONSTANT
:300 : #1F0000=6D : CONSTANT
:301 : #200001=6E : CONSTANT
:302 : #F27000=6F : CONSTANT
:303 : #3000000=70 : CONSTANT
:304 : #41F0000=71 : CONSTANT
:305 : #7000000=72 : CONSTANT
:306 : #3FFFFFFE00=73 : CONSTANT
:307 : #7F800000=74 : CONSTANT
:308 : #7FFFFFFE00=75 : CONSTANT
:309 : #7FFFFFF0E=76 : CONSTANT
:310 : #BF800001=77 : CONSTANT
:311 :
:312 : GPR.OS=78 : HARDWARE INDEX TO GPRS
:313 : BIT.OS(3-0)=79 : 16 LOCATION HARDWARE INDEX TO BIT MASKS
:314 : BIT.OS(4-0)=7B : 32 LOCATION HARDWARE INDEX TO BIT MASKS
:315 : OS=7C : OS REGISTER
:316 : DEC.CON=7D : DECIMAL CARRIES
:317 : SIZE=7E : SIZE REGISTER
:318 : MDT= 7E : MEMORY REFERENCE DATA SIZE
:319 : INTERRUPT.VEC 7E : HARDWARE INTERRUPT VECTOR

```

```

320 :
321 : THIS WORD IS THE HARDWARE CONDITION CODES. THE CC'S CAN BE READ
322 : OR ARE WRITTEN HERE. NO OTHER BITS IN THE PSL ARE EFFECTED.
323 :
324 : PSL.CC=7F ; PSL CONDITION CODES
325 :
326 : XD.ADRS/=<16:9>,.VALIDITY=<XD.VAL>
327 :
328 : SAVED.2ND.REQUEST=0 ; MM SAVED STATE FOR REQUEST.
329 : T1=1 ; TEMP LOCATION 1
330 : T2=2 ; TEMP LOCATION 2
331 : T3=3 ; TEMP LOCATION 3
332 : T4=4 ; TEMP LOCATION 4
333 : T5=5 ; TEMP LOCATION 5
334 : T6=6 ; TEMP LOCATION 6
335 : T7=7 ; TEMP LOCATION 7
336 : T8=8 ; TEMP LOCATION 8
337 : T9=9 ; TEMP LOCATION 9
338 : T10=0A ; TEMP LOCATION 10
339 : BACKUP.CM.PC=0A ; COMPATIBILITY MODE KEEPS ORIGINAL PC HERE.
340 : T11=0B ; TEMP LOCATION 11
341 : T12=0C ; TEMP LOCATION 12
342 : T13=0D ; TEMP LOCATION 13
343 : T14=0E ; TEMP LOCATION 14
344 : T15=0F ; TEMP LOCATION 15
345 : PC=10 ; MACRO INSTRUCTION PC
346 : WR.BACKUP=11 ; BACKUP WORKING REGISTER FOR MOV MEM.DATA TO WR[].
347 : SAVED.VAR=12 ; MM SAVED VAR LOCATION
348 : SAVED.DATA=13 ; MM SAVED DATA LOCATION
349 : SAVED.PTE.ADDRESS=14 ; MM SAVED PAGE TABLE ENTRY ADDRESS
350 : SAVED.PTE=15 ; MM SAVED PAGE TABLE ENTRY
351 : T16=16 ; TEMP LOCATION 16
352 : T17=17 ; TEMP LOCATION 17
353 : IE.TEMP4=18 ; INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION FOUR.
354 : #FFFFFF00=19 ; CONSTANT
355 : #FFFFFF00=1A ; CONSTANT
356 : #FFFFFFFC=1B ; CONSTANT
357 : #FFFFFFF=1C ; CONSTANT
358 :
359 : THIS LOCATION CONTAINS THE SOFT COPY OF THE PSL. WHENEVER THE HARDWARE
360 : VERSION OF THE PSL IS CHANGED (PSL.HW) THEN THIS CHANGE IS
361 : REPORTED HERE. ALL BITS OF THE PSL ARE REALLY LIKE THOSE HERE
362 : EXCEPT FOR THE CONDITION CODES WHICH MUST BE OBTAINED
363 : FROM PSL.CC.
364 :
365 : PSL=1D ; SOFT COPY OF VAX PSL
366 : VAR=1E ; VIRTUAL ADDRESS REGISTER
367 : VAR2=1F ; VIRTUAL ADDRESS REGISTER
368 :
369 :
370 : #1=20 ; MASK 00000001 (HEX)
371 : #2=21 ; MASK 00000002
372 : #4=22 ; MASK 00000004
373 : #8=23 ; MASK 00000008
374 : #10=24 ; MASK 00000010
  
```

22-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) FIELD DEFINITIONS F 3-MAR-82
FIELD DEFINITIONS FOR MAIN MICRO WORD

C 2

3-MAR-82 10:02:46

ENKCB

Fiche 1 Frame C2
Micro-diagnostic for DAP module
VER 00.08

Sequence 15
Rev 01.00

Page 9

:375	:	#20=25	:	MASK	00000020
:376	:	#40=26	:	MASK	00000040
:377	:	#80=27	:	MASK	00000080
:378	:	#100=28	:	MASK	00000100
:379	:	#200=29	:	MASK	00000200
:380	:	#400=2A	:	MASK	00000400
:381	:	#800=2B	:	MASK	00000800
:382	:	#1000=2C	:	MASK	00001000
:383	:	#2000=2D	:	MASK	00002000
:384	:	#4000=2E	:	MASK	00004000
:385	:	#8000=2F	:	MASK	00008000
:386	:	#10000=30	:	MASK	00010000
:387	:	#20000=31	:	MASK	00020000
:388	:	#40000=32	:	MASK	00040000
:389	:	#80000=33	:	MASK	00080000
:390	:	#100000=34	:	MASK	00100000
:391	:	#200000=35	:	MASK	00200000
:392	:	#400000=36	:	MASK	00400000
:393	:	#800000=37	:	MASK	00800000
:394	:	#1000000=38	:	MASK	01000000
:395	:	#2000000=39	:	MASK	02000000
:396	:	#4000000=3A	:	MASK	04000000
:397	:	#8000000=3B	:	MASK	08000000
:398	:	#10000000=3C	:	MASK	10000000
:399	:	#20000000=3D	:	MASK	20000000
:400	:	#40000000=3E	:	MASK	40000000
:401	:	#80000000=3F	:	MASK	80000000
:402	:		:		
:403	:	BIT0=20	:	MASK	00000001
:404	:	BIT1=21	:	MASK	00000002
:405	:	BIT2=22	:	MASK	00000004
:406	:	BIT3=23	:	MASK	00000008
:407	:	BIT4=24	:	MASK	00000010
:408	:	BIT5=25	:	MASK	00000020
:409	:	BIT6=26	:	MASK	00000040
:410	:	BIT7=27	:	MASK	00000080
:411	:	BIT8=28	:	MASK	00000100
:412	:	BIT9=29	:	MASK	00000200
:413	:	BIT10=2A	:	MASK	00000400
:414	:	BIT11=2B	:	MASK	00000800
:415	:	BIT12=2C	:	MASK	00001000
:416	:	BIT13=2D	:	MASK	00002000
:417	:	BIT14=2E	:	MASK	00004000
:418	:	BIT15=2F	:	MASK	00008000
:419	:	BIT16=30	:	MASK	00010000
:420	:	BIT17=31	:	MASK	00020000
:421	:	BIT18=32	:	MASK	00040000
:422	:	BIT19=33	:	MASK	00080000
:423	:	BIT20=34	:	MASK	00100000
:424	:	BIT21=35	:	MASK	00200000
:425	:	BIT22=36	:	MASK	00400000
:426	:	BIT23=37	:	MASK	00800000
:427	:	BIT24=38	:	MASK	01000000
:428	:	BIT25=39	:	MASK	02000000
:429	:	BIT26=3A	:	MASK	04000000

430	:	BIT27=3B	:	MASK 08000000
431	:	BIT28=3C	:	MASK 10000000
432	:	BIT29=3D	:	MASK 20000000
433	:	BIT30=3E	:	MASK 40000000
434	:	BIT31=3F	:	MASK 80000000
435	:			
436	:	GPR0=40	:	GPR 0
437	:	GPR1=41	:	GPR 1
438	:	GPR2=42	:	GPR 2
439	:	GPR3=43	:	GPR 3
440	:	GPR4=44	:	GPR 4
441	:	GPR5=45	:	GPR 5
442	:	GPR6=46	:	GPR 6
443	:	CM.SP=46	:	IN COMPATIBILITY MODE THIS IS THE STACK POINTER.
444	:	GPR7=47	:	GPR 7
445	:	CM.PC=47	:	IN COMPATIBILITY MODE THIS IS THE PC.
446	:	GPR8=48	:	GPR 8
447	:	GPR9=49	:	GPR 9
448	:	GPR10=4A	:	GPR 10
449	:	GPR11=4B	:	GPR 11
450	:	GPR12=4C	:	GPR 12
451	:	AP=4C	:	ARGUMENT POINTER
452	:	GPR13=4D	:	GPR 13
453	:	FP=4D	:	FRAME POINTER
454	:	SP=4E	:	GPR 14
455	:	TO=4F	:	TEMP LOCATION 0
456	:			
457	:	ZERO=50	:	ZERO CONSTANT
458	:	#3=51	:	CONSTANT
459	:	#5=52	:	CONSTANT
460	:	#6=53	:	CONSTANT
461	:	#7=54	:	CONSTANT
462	:	#9=55	:	CONSTANT
463	:	#8=56	:	CONSTANT
464	:	#C=57	:	CONSTANT
465	:	#D=58	:	CONSTANT
466	:	#E=59	:	CONSTANT
467	:	#F=5A	:	CONSTANT
468	:	#13=5B	:	CONSTANT
469	:	#1F=5C	:	CONSTANT
470	:	#34=5D	:	CONSTANT
471	:	#3B=5E	:	CONSTANT
472	:	#3F=5F	:	CONSTANT
473	:	#4B=60	:	CONSTANT
474	:	#66=61	:	CONSTANT
475	:	#A0=62	:	CONSTANT
476	:	#F0=63	:	CONSTANT
477	:	#FF=64	:	CONSTANT
478	:	#1FF=65	:	CONSTANT
479	:	#FFF=66	:	CONSTANT
480	:	#2001=67	:	CONSTANT
481	:	#3000=68	:	CONSTANT
482	:	#7F80=69	:	CONSTANT
483	:	#C000=6A	:	CONSTANT
484	:	#FFE0=6B	:	CONSTANT

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

FIELD DEFINITIONS F 3-MAR-82
MICRO2 1L(03) 3-MAR-82 10:02:46
FIELD DEFINITIONS FOR MAIN MICRO WORD

E 2

Fiche 1 Frame E2

Sequence 17

ENKCB Micro-diagnostic for DAP module
VER 00.08

Rev 01.00

Page 11

```
:485 : #FFFF=6C : CONSTANT
:486 : #1F0000=6D : CONSTANT
:487 : #200001=6E : CONSTANT
:488 : #F27000=6F : CONSTANT
:489 : #3000000=70 : CONSTANT
:490 : #41F0000=71 : CONSTANT
:491 : #7000000=72 : CONSTANT
:492 : #3FFFFFF0=73 : CONSTANT
:493 : #7F800000=74 : CONSTANT
:494 : #7FFFFFF0=75 : CONSTANT
:495 : #7FFFFFF0E=76 : CONSTANT
:496 : #BF800001=77 : CONSTANT
:497 :
:498 : GPR.OS=78 : HARDWARE INDEX TO GPRS
:499 : BIT.OS(3-0)=79 : 16 LOCATION HARDWARE INDEX TO BIT MASKS
:500 : BIT.OS(4-0)=7B : 32 LOCATION HARDWARE INDEX TO BIT MASKS
:501 : OS=7C : OS REGISTER
:502 : DEC.CON=7D : DECIMAL CARRIES
:503 : SIZE=7E : SIZE REGISTER
:504 : MDT= 7E : MEMORY REFERENCE DATA SIZE
:505 : INTERRUPT.VEC=7E : HARDWARE INTERRUPT VECTOR
:506 :
:507 : THIS LOCATION IS THE HARDWARE CONDITION CODES. THE CC'S CAN BE
:508 : READ OR ARE WRITTEN HERE. NO OTHER BITS IN THE PSL ARE EFFECTED.
:509 :
:510 : PSL.CC=7F : PSL CONDITION CODES
:511 :
:512 : THESE ADDRESSES ARE ONLY ACCESSIBLE WITH THE MOV MICRO INSTRUCTION.
:513 :
:514 : KSP=80 : KERNEL STACK POINTER
:515 : ESP=81 : EXECUTIVE STACK POINTER
:516 : SSP=82 : SUPERVISOR STACK POINTER
:517 : USP=83 : USER STACK POINTER
:518 : ISP=84 : INTERRUPT STACK POINTER
:519 : ASTLVL=85 : AST LEVEL
:520 : PCBB=86 : PROCESS CONTROL BLOCK BASE
:521 : SCBB=87 : SYSTEM CONTROL BLOCK BASE
:522 : SISR=88 : SOFTWARE INTERRUPT SUMMARY REGISTER
:523 :
:524 : CONSOLE.COMMAND=89 : ONE BYTE COMMAND.
:525 : CONSOLE.MODIFIER=8A : ONE BYTE MODIFIER.
:526 : CONSOLE.ADDRESS=8B : FOUR BYTES OF ADDRESS.
:527 : CONSOLE.DATA=8C : FOUR BYTES OF DATA.
:528 : CONSOLE.STATUS=8D : ONE BYTE OF STATUS.
:529 : PSEUDO.NICR=8E : COPY OF NICR THAT 8085 HAS.
:530 : PSEUDO.ICR=8F : COPY OF ICR THAT 8085 HAS.
:531 : DEBUG.SAVE.WR0=90 : TEMPORARY SAVE WR0.
:532 : DEBUG.SAVE.WR1=91 : TEMPORARY SAVE WR1.
:533 : DEBUG.SAVE.ALU.CC=92 : TEMPORARY SAVE CONDITION CODES.
:534 : MEMORY.SIZE=93 : MEMORY SIZE OF THE MACHINE.
:535 : CRD.LOG=94 : INFORMATION ABOUT LAST SOFT MEMORY ERROR.
:536 : PACKET.A=95 : TOP OF PACKET.
:537 : PACKET.B=96 : BOTTOM OF PACKET.
:538 : DEBUG.SAVE.ADDRESS=97 : ADDRESS FOR EXAMINE/DEPOSIT.
:539 :
```

:540 : SAVED,MDT=99 : SAVED ADDRESS OF MDT FOR MMIE OPTIMIZATION.
 :541 : POBR=9A : PO BASE REGISTER
 :542 : POLR=9B : PO LENGTH REGISTER
 :543 : P1BR=9C : P1 BASE REGISTER
 :544 : P1LR=9D : P1 LENGTH REGISTER
 :545 : SBR=9E : SYSTEM BASE REGISTER
 :546 : SLR=9F : SYSTEM LENGTH REGISTER

:548 : PORT0=0A0 : RESERVED FOR PORT.
 :549 : PORT1=0A1 : RESERVED FOR PORT.
 :550 : PORT2=0A2 : RESERVED FOR PORT.
 :551 : PORT3=0A3 : RESERVED FOR PORT.
 :552 : PORT4=0A4 : RESERVED FOR PORT.
 :553 : PORT5=0A5 : RESERVED FOR PORT.
 :554 : PORT6=0A6 : RESERVED FOR PORT.
 :555 : PORT7=0A7 : RESERVED FOR PORT.
 :556 : PORT8=0A8 : RESERVED FOR PORT.
 :557 : PORT9=0A9 : RESERVED FOR PORT.
 :558 : PORT10=0AA : RESERVED FOR PORT.
 :559 : PORT11=0AB : RESERVED FOR PORT.
 :560 : PORT12=0AC : RESERVED FOR PORT.
 :561 : PORT13=0AD : RESERVED FOR PORT.
 :562 : PORT14=0AE : RESERVED FOR PORT.
 :563 : PORT15=0AF : RESERVED FOR PORT.
 :564 : PORT16=0B0 : RESERVED FOR PORT.
 :565 : PORT17=0B1 : RESERVED FOR PORT.
 :566 : PORT18=0B2 : RESERVED FOR PORT.
 :567 : PORT19=0B3 : RESERVED FOR PORT.
 :568 : PORT20=0B4 : RESERVED FOR PORT.
 :569 : PORT21=0B5 : RESERVED FOR PORT.
 :570 : PORT22=0B6 : RESERVED FOR PORT.
 :571 : PORT23=0B7 : RESERVED FOR PORT.
 :572 : PORT24=0B8 : RESERVED FOR PORT.
 :573 : PORT25=0B9 : RESERVED FOR PORT.
 :574 : PORT26=0BA : RESERVED FOR PORT.
 :575 : PORT27=0BB : RESERVED FOR PORT.
 :576 : PORT28=0BC : RESERVED FOR PORT.
 :577 : PORT29=0BD : RESERVED FOR PORT.
 :578 : PORT30=0BE : RESERVED FOR PORT.
 :579 : PORT31=0BF : RESERVED FOR PORT.

:580 :
 :581 : XGPR0=0C0 : BACKUP COPY OF GPR 0
 :582 : XGPR1=0C1 : BACKUP COPY OF GPR 1
 :583 : XGPR2=0C2 : BACKUP COPY OF GPR 2
 :584 : XGPR3=0C3 : BACKUP COPY OF GPR 3
 :585 : XGPR4=0C4 : BACKUP COPY OF GPR 4
 :586 : XGPR5=0C5 : BACKUP COPY OF GPR 5
 :587 : XGPR6=0C6 : BACKUP COPY OF GPR 6
 :588 : XGPR7=0C7 : BACKUP COPY OF GPR 7
 :589 : XGPR8=0C8 : BACKUP COPY OF GPR 8
 :590 : XGPR9=0C9 : BACKUP COPY OF GPR 9
 :591 : XGPR10=0CA : BACKUP COPY OF GPR 10
 :592 : XGPR11=0CB : BACKUP COPY OF GPR 11
 :593 : XGPR12=0CC : BACKUP COPY OF GPR 12
 :594 : XGPR13=0CD : BACKUP COPY OF GPR 13

```

:595 : XSP=0CE : BACKUP COPY OF GPR 14
:596 :
:597 : #14=0D0 : CONSTANT
:598 : #18=0D1 : CONSTANT
:599 : #1C=0D2 : CONSTANT
:600 : #24=0D3 : CONSTANT
:601 : #28=0D4 : CONSTANT
:602 : #2C=0D5 : CONSTANT
:603 : #2D=0D6 : CONSTANT
:604 : #30=0D7 : CONSTANT
:605 : #F8=0D8 : CONSTANT
:606 : #FC=0D9 : CONSTANT
:607 : #2D20=0DA : CONSTANT
:608 : #140000=0DB : CONSTANT
:609 : #150000=0DC : CONSTANT
:610 : #160000=0DD : CONSTANT
:611 : #170000=0DE : CONSTANT
:612 : #180000=0DF : CONSTANT
:613 : #1E0000=0E0 : CONSTANT
:614 : #40A10=0E1 : CONSTANT
:615 : #C0000E0=0E2 : CONSTANT
:616 : #0FFF0000=0E3 : CONSTANT
:617 : #B020FF00=0E4 : CONSTANT
:618 : #FFFFFF10=0E5 : CONSTANT
:619 : DEBUG.SAVE.T1=0E6 : TEMPORARY SAVE T1.
:620 : DEBUG.SAVE.OS=0E7 : TEMPORARY SAVE OS.
:621 : DEBUG.SAVE.SIZE=0E8 : TEMPORARY SAVE SIZE VALUE.
:622 : SAVED.WR0=0E9 : MM SAVED WR0
:623 : SAVED.WR1=0EA : MM SAVED WR1
:624 : SAVED.2ND.VAR=0EB : MM SAVED VAR 2
:625 : SAVED.ALU.CC=0EC : MM SAVED ALU CONDITION CODES
:626 : SAVED.SIZE=0ED : MM SAVED LS[SIZE]
:627 : SAVED.OS=0EE : MM SAVED LS[OS]
:628 : SAVED.REQUEST=0EF : MM SAVED REQUEST DATA
:629 : SAVED.CRD.LOG=0F0 : MM SAVED LOCATION
:630 : OS.BAK=0F1 : USED BY WARM FLOATING POINT.
:631 :
:632 : THIS LOCATION IS FOR USE BY THE MFPR AND MTPR TO THE CONSOLE AND
:633 : TUSB CSR'S. ALL OF THE INTERRUPT ENABLE BITS ARE HEREIN.
:634 :
:635 : CONSOLE/PROGRAM I/O MODE FLAG - BIT<31> 0 - CONSOLE 1 - PROGRAM I/O MODE
:636 :
:637 : CRD RETRY IN PROGRESS - BIT<5> 0 - NOTHING DOING 1 - IN PROGRESS
:638 : MACHINE CHECK IN PROGRESS - BIT<4>
:639 : TUSB TRANSMIT CSR IE - BIT<3>
:640 : TUSB RECEIVER CSR IE - BIT<2>
:641 : CONSOLE TRANSMIT CSR IE - BIT<1>
:642 : CONSOLE TRANSMIT CSR IE - BIT<0>
:643 :
:644 : CONSOLE.CSR.IES=0F2 : CONSOLE CSR IE'S AND CONSOLE/PROGRAM I/O MODE FLAG.
:645 : IE.TEMP1=0F3 : INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION ONE
:646 : IE.TEMP2=0F4 : INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION TWO
:647 : IE.TEMP3=0F5 : INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION THREE
:648 :
:649 : XGPR.OS=0F8 : HARDWARE INDEX TO XGPRS
  
```

:650 : PORT(3-0)=0F9 : 16 LOCATION HARDWARE INDEX TO PORT REGISTERS.
 :651 : PORT(4-0)=0FB : 32 LOCATION HARDWARE INDEX TO PORT REGISTERS.
 :652 : CCR=0FC : CONSOLE READ REGISTER
 :653 : TIMER=0FD : INTERVAL TIMER VALUE.
 :654 : ALU.CC=0FE : ALU CONDITION CODES

:655 :
 :656 : THIS LOCATION IS A WRITE ONLY REGISTER. THE FOLLOWING BITS IN THE
 :657 : PSL ARE AFFECTED:

:658 :
 :659 : COMPATIBILITY MODE - BIT<31>
 :660 : CURRENT MODE - BITS<25:24>
 :661 : INTERRUPT PRIORITY LEVEL (IPL) - BITS<20:16>
 :662 : TRACE TRAP ENABLE (REALLY T OR TP) - BIT<4>
 :663 : NEGATIVE CONDITION CODE - BIT<3>
 :664 : ZERO CONDITION CODE - BIT<2>
 :665 : OVERFLOW CONDITION CODE - BIT<1>
 :666 : CARRY CONDITION CODE - BIT<0>

:667 :
 :668 : PSL.HW=OFF : HARDWARE PSL BITS

:669 :
 :670 : DATA PATH CONTROL

:671 :
 :672 : THIS FIELD IS BUILT FROM AN ADDRESS IN THE CCODE MEMORY. THE CCODE MEMORY
 :673 : CONTAINS DATA PATH MICRO INSTRUCTIONS TO PRODUCE THE DESIRED OPERATION
 :674 : FOR THE UCODE MACROS.

:675 : DATA PATH CONTROL FIELD FOR THE 'BASIC' MICRO INSTRUCTION

:676 :
 :677 : DP/= <21:16>, .VALIDITY=<DP.VAL>, .DEFAULT=<.SHIFT[.AND[<SDP/Q.CMP.LS>,OFF],-2]>

:678 :
 :679 : THIS FIELD IS ONLY USED WITH THE INSTRUCTIONS WHICH CAN HAVE
 :680 : XCHG ATTACHED.

:681 : DP.SMALL/= <20:16>, .VALIDITY=<DP.VAL>

:682 : EXCHANGE ORIGINAL WR TO LS

:683 : XCHG.BIT/= <21>, .DEFAULT=<0>

:684 : YES=1
 :685 : NO=0

:686 : DATA PATH CONTROL FIELD FOR THE 'EXTENDED' MICRO INSTRUCTION

:687 : XDP/= <19:14>, .VALIDITY=<A.VAL>

:688 : DATA PATH CONTROL FOR THE 'MOVE' MICRO INSTRUCTION

:689 : MDP/= <19:17>, .VALIDITY=<XD.VAL>

:690 :

```

:701 .PAGE
:702 : CONDITION CODE FIELD / DATA PATH CONTEXT
:703
:704 CC/= <6:5>, .VALIDITY=<CC.VAL>, .DEFAULT=<CC/DT(LONG)&HOLD.ALU.CC>
:705
:706 DT(LONG)&HOLD.ALU.CC=0 ; USE LONG CONTEXT(32 BITS) AND HOLD ALU CONDITION CODES
:707 DT(LONG)&SET.ALU.CC=1 ; USE LONG CONTEXT(32 BITS) AND SET ALU CONDITION CODES
:708 DT(SIZE)&SET.ALU.CC=2 ; USE CONTEXT IN THE SIZE REGISTER AND SET ALU CONDITION CODES
:709 DT(SIZE)&MAP.ALU.CC.TO.PSL=3 ; TRANSFER ALU CONDITION CODES TO PSL CONDITION CODES HARDWARE DEPENDENT
:710
:711
:712 : MEMORY DATA TYPE FIELD
:713
:714 MDT/= <6:5>, .VALIDITY=<DT.VAL>
:715
:716 BYTE=0 ; SIZE = BYTE
:717 WORD=1 ; SIZE = WORD
:718 SIZE=2 ; SIZE = CONTENTS OF SIZE REGISTER
:719 LONG=3 ; SIZE = LONG
:720
:721
:722 : SHIFT INPUT FIELD
:723
:724 : THIS CONTROL FIELD DETERMINES THE SHIFT INPUTS FOR THE FOLLOWING OPERATIONS.
:725 : THE DIRECTION OF THE SHIFT IS DETERMINED BY THE 2901 DESTINATION CODE.
:726
:727 SI/= <13:11>, .VALIDITY=<A.VAL>
:728
:729 ROT.WR=0 ; ROTATE WR
:730 ROT.WR.Q=1 ; ROTATE WR AND Q
:731 ASH.WR=2 ; ARITHMETIC SHIFT OF WR
:732 ASH.WR.Q=3 ; ARITHMETIC SHIFT OF WR AND Q
:733 MUL.SHF=4 ; SIGNED MULTIPLY SHIFT OPERATION
:734 UMUL.SHF=5 ; UNSIGNED MULTIPLY SHIFT OPERATION
:735 SHF.WR.Q=6 ; LOGICAL SHIFT WR AND Q
:736
:737
:738 : SKIP MICRO CONTROL FIELD
:739
:740 : THIS FIELD IS VALID FOR ALL INSTRUCTIONS EXCEPT
:741
:742 JUMP MICRO INSTRUCTION
:743 DECODE MICRO INSTRUCTION
:744
:745 SCTL/= <4:0>, .VALIDITY=<.AND[<SCTL.VAL>, <SKIP.PAGE.VAL>]>, .DEFAULT=<SCTL/NO.SKIP>
:746
:747 N.CLR=0 ; ALU N-BIT EQUAL ZERO
:748 NEQ=1 ; ALU Z-BIT EQUAL ZERO
:749 BIT.SET=1 ; ALU Z-BIT EQUAL ZERO
:750 V.CLR=2 ; ALU V-BIT EQUAL ZERO
:751 C.SET=3 ; ALU C-BIT EQUAL ONE
:752 GEQU=3 ; ALU C-BIT EQUAL ONE
:753 NOT.PSL.C=4 ; PSL C EQUAL ZERO
:754 BR.FALSE=5 ; BRANCH INSTRUCTION FALSE
:755 R.DST=6 ; REGISTER MODE FLAG
    
```

```

:756 NO.INTERRUPT=7 : NO INTERRUPT PENDING
:757 N.SET=8 : ALU N-BIT EQUAL ONE
:758 EQL=9 : ALU Z-BIT EQUAL ONE
:759 BITS.CLR=9 : ALU Z-BIT EQUAL ONE
:760 V.SET=0A : ALU V-BIT EQUAL ONE
:761 C.CLR=0B : ALU C-BIT EQUAL ZERO
:762 LSSU=0B : ALU C-BIT EQUAL ZERO
:763 STATE.ZERO.SET=0C : STATE REGISTER BIT ZERO TRUE
:764 STATE.ONE.SET=0D : STATE REGISTER BIT ONE TRUE
:765 STATE.ZERO.CLR=0E : STATE REGISTER BIT ZERO FALSE
:766 STATE.ONE.CLR=0F : STATE REGISTER BIT ONE FALSE
:767 RET.NOERR=10 : RETURN+1 IF NO MEMORY ERROR
:768 JMP.Z.CLR=11 : JUMP TO (STACK) IF ALU Z = 0
:769 POP.USTACK=12 : DISCARD TOP STACK ENTRY
:770 JMP.C.SET=13 : JUMP TO (STACK) IF ALU C=1
:771 RETURN=14 : RETURN TO (USTACK)
:772 NO.SKIP=15 : EXECUTE NEXT INSTRUCTION
:773 RETURN+1=16 : RETURN TO (USTACK)+1
:774 LOOP.IF.NO.1.AND.SYNC=17 : JUMP TO (STACK) IF NO INTERRUPT AND NO ACCELERATOR SYNC.
:775 CONSOLE.ATTN=18 : CONSOLE ATTENTION
:776 CONSOLE.ACK=19 : CONSOLE ACKNOWLEDGE
:777 NO.FAST.INTERRUPT=1A : NO FAST INTERRUPTS PENDING
:778 BACKUP.MASK.VALID=1B : REGISTER BACKUP MASK VALID.
:779 LSS=1C : SIGNED LESS THAN.
:780 MEM.REF.OK=1D : NO MEMORY ERROR TRUE
:781 SKIP=1E : EXECUTE ADDRESS PLUS ONE
:782 SYNC=1F : ACCELERATOR SYNCHRONIZATION
:783
:784

```

: JUMP MICRO CONTROL FIELD

: THIS FIELD IS VALID FOR THE FOLLOWING INSTRUCTIONS

: JUMP MICRO INSTRUCTION
: DECODE MICRO INSTRUCTION

: JCTL/= <3:0>, .VALIDITY= <.AND[<JCTL.VAL>, <JUMP.PAGE.VAL>]>, .DEFAULT= <JCTL/JUMP.VALID>

```

:794 N.CLR=0 : ALU N-BIT EQUAL ZERO
:795 NEQ=1 : ALU Z-BIT EQUAL ZERO
:796 BIT.SET=1 : ALU Z-BIT EQUAL ZERO
:797 V.CLR=2 : ALU V-BIT EQUAL ZERO
:798 C.SET=3 : ALU C-BIT EQUAL ONE
:799 GEQU=3 : ALU C-BIT EQUAL ONE
:800 JUMP=4 : JUMP UNCONDITIONALLY
:801 BR.FALSE=5 : BRANCH INSTRUCTION FALSE
:802 R.DST=6 : REGISTER MODE FLAG
:803 NO.INTERRUPT=7 : NO INTERRUPT PENDING
:804 N.SET=8 : ALU N-BIT EQUAL ONE
:805 EQL=9 : ALU Z-BIT EQUAL ONE
:806 BITS.CLR=9 : ALU Z-BIT EQUAL ONE
:807 V.SET=0A : ALU V-BIT EQUAL ONE
:808 C.CLR=0B : ALU C-BIT EQUAL ZERO
:809 LSSU=0B : ALU C-BIT EQUAL ZERO
:810 JSR=0C : UNCONDITIONAL JUMP AND PUSH USTACK

```



```

:811      JSR.VALID=0D      ; JUMP AND PUSH USTACK IF IB VALID=1
:812      NO.JUMP.TST=0E   ; DO NOT JUMP AND SKIP IF IB VALID=0
:813      JUMP.VALID=0F    ; JUMP IF IB VALID=1
:814
:815
:816      ; JUMP ADDRESS FIELD
:817
:818      JUMP.ADRS/=<17:4>,.ADDRESS,.VALIDITY=<JUMP.VAL>
:819
:820
:821      ; OS ENABLE
:822
:823      OS/=<18>,.VALIDITY=<JUMP.VAL>
:824
:825      NOP=0
:826      WITH.OS=1        ; CONCATENATE OS REGISTER TO JUMP ADDRESS
:827
:828
  
```

```

:829 .PAGE
:830 : BACKUP PC
:831
:832 BPC/= <18>, .VALIDITY=<DECODE.VAL>, .DEFAULT=<.CASE[<.EQL[<OPC2/>, <OPC2/DECODE>]]>JOF[0,1]>
:833
:834 NOP=1
:835 SAVE.PC=0 ; WRITE ALU DATA INTO WR[B] (PC+1)
:836
:837
:838 : DECODE ADDRESS FIELD
:839
:840 DEC.ADRS/= <17:12>, .VALIDITY=<DECODE.VAL>
:841
:842
:843 : IR FUNCTION
:844
:845 IFUNC/= <11:9>, .VALIDITY=<DECODE.VAL>
:846
:847 CM.EXEC=0 ; COMPATIBILITY MODE EXECUTION DISPATCH WHEN UPPER BYTE = ZERO
:848 SPEC=0 ; SPECIFIER DISPATCH
:849 CM.IRD=1 ; COMPATIBILITY MODE INSTRUCTION DECODE DISPATCH
:850 FLOAT=1 ; SPECIFIER FLOAT TYPE DISPATCH
:851 VAX.EXEC=2 ; VAX EXECUTION DISPATCH
:852 ASRC=2 ; ADDRESS SPECIFIER DISPATCH
:853 VAX.IRD=3 ; VAX OPCODE DISPATCH
:854 VSRC=4 ; ADDRESS SPECIFIER DISPATCH FOR BIT FIELD INSTRUCTIONS
:855 ESRC=5 ; SPECIFIER DISPATCH IN INDEX MODE
:856 CM.DST=6 ; COMPATIBILITY MODE DESTINATION DISPATCH
:857 CM.SINGLE=7 ; COMPATIBILITY MODE SOP DISPATCH
:858
:859
:860 : INSTRUCTION BUFFER REQUEST
:861
:862 IB.REQ/= <8>, .VALIDITY=<DECODE.VAL>, .DEFAULT=0
:863
:864 NOP=0
:865 IB.REQ=1 ; ENABLE HARDWARE DEPENDENT MEMORY REQUEST
:866
:867 : COMPATIBILITY MODE OPCODE SELECT
:868
:869 CM.HI/= <7>, .VALIDITY=<DECODE.VAL>, .DEFAULT=0
:870
:871 LOW.BYTE=0 ; DECODE IR BITS<7:0>
:872 HI.BYTE=1 ; DECODE IR BITS <15:6>
:873
:874
:875 : LOAD OS REGISTER
:876
:877 LD.OS/= <6>, .VALIDITY=<DECODE.VAL>, .DEFAULT=0
:878
:879 NOP=0
:880 LOAD.OS=1 ; LOAD OS REGISTER FROM I-BUFFER
:881
:882
:883 : REGISTER DESTINATION FLAG ENABLE
    
```

```

:884 ;
:885 R.DST/= <5> ,.VALIDITY=<DECODE.VAL> ,.DEFAULT=0
:886
:887 NOP=0
:888 ENAB=1
:889 ;
:890 ; OPCODE SPECIFIER BIT
:891 ;
:892 OPC.SPEC/= <4> ,.VALIDITY=<DECODE.VAL>
:893
:894 CM.EXEC=1 ; COMPATIBILITY MODE EXECUTION DISPATCH
:895 SPEC=0 ; SPECIFIER DISPATCH
:896 CM.IRD=1 ; COMPATIBILITY MODE INSTRUCTION DECODE DISPATCH
:897 FLOAT=0 ; SPECIFIER FLOAT TYPE DISPATCH
:898 VAX.EXEC=1 ; VAX EXECUTION DISPATCH
:899 ASRC=0 ; ADDRESS SPECIFIER DISPATCH
:900 VAX.IRD=1 ; VAX OPCODE DISPATCH
:901 VSRC=0 ; ADDRESS SPECIFIER DISPATCH FOR BIT FIELD INSTRUCTIONS
:902 ESRC=0 ; SPECIFIER DISPATCH IN INDEX MODE
:903 CM.DST=0 ; COMPATIBILITY MODE DESTINATION DISPATCH
:904 CM.SINGLE=0 ; COMPATIBILITY MODE SOP DISPATCH
  
```

```

:905 .PAGE
:906 . MISCELLANEOUS FUNCTIONS
:907
:908
:909 . THIS FIELD IS A FLAG FOR STEERING THE MICROWORD AS A MISCELLANEOUS
:910 . INSTRUCTION OR AS A PORT INSTRUCTION.
:911
:912 MISC.PORT/= <18>
:913
:914     MISC=0
:915     PORT=1
:916
:917 MISC.0/= <17>,.VALIDITY=<MISC.VAL>
:918
:919     NOP=0 ; NO OPERATION IN ALU OR CONDITION CODES.
:920
:921 MISC.1/= <15:12>,.VALIDITY=<MISC.VAL>
:922
:923     NOP=0F ; NO OPERATION
:924     SET.CP.ATTN=0E ; SET CPU ATTENTION FOR CONSOLE
:925     SET.CP.ACK=0D ; SET CPU ACKNOWLEDGE FOR CONSOLE
:926     CLR.CP.ATTN.AND.ACK=0C ; CLEAR CPU ATTENTION AND CPU ACKNOWLEDGE
:927     SET.HIGH.PAGE=0B ; SET BIT FOR HIGH HALF OF WCS.
:928     CLR.HIGH.PAGE=0A ; CLEAR BIT FOR LOW HALF OF WCS.
:929     SET.PORT.I.O=9 ; SET PORT I/O MODE
:930     SET.ACC.I.O=8 ; SET ACCELERATOR I/O MODE
:931     SET.BACKUP.MASK.VALID=7 ; SET REGISTER BACKUP MASK FLAG
:932     SET.S1=6 ; SET STATE ONE BIT
:933     SET.S0=5 ; SET STATE ZERO BIT
:934     CLR.S1=4 ; CLEAR STATE ONE BIT
:935     CLR.S0=3 ; CLEAR STATE ZERO BIT
:936     SET.S1&CLR.S0=2 ; SET STATE ONE BIT AND CLEAR STATE ZERO BIT
:937     CLR.S1&SET.S0=1 ; CLEAR STATE ONE AND SET STATE ZERO
:938     CLR=0 ; CLEAR STATE ONE AND STATE ZERO BITS.
:939
:940
:941 MISC.2/= <11:9>,.VALIDITY=<MISC.VAL>
:942
:943     NOP=0 ; NO OPERATION
:944     MASK.INTERRUPTS=1 ; MASK ALL INTERRUPTS (FOR DECODE NEXT CYCLE).
:945     ASSERT.DA.AND.PASS.WR=2 ; ASSERT DATA AVAILABLE AND PASS WR[0] TO THE ACCELERATOR OR PORT.
:946     TRAP.ACC=3 ; TRAP ACCELERATOR
:947     READ.ACC.UPC=4 ; READ ACCELERATOR PROGRAM COUNTER
:948     TRANSFER.GRANT=5 ; RECOGNIZE PORT REQUEST.
:949     MASK.HALT.AND.T.BIT=6 ; MASK HALT AND T-BIT TRAPS (FOR DECODE TWO CYCLES LATER)
:950     WRITE.WR.TO.CONSOLE.REGISTER=7 ; SEND BITS <7:0> TO 8085.
:951
:952 MISC.WR/= <8:7>
:953
:954 MISC.WRL/= <8>
:955 MISC.WRR/= <7>
:956
:957 PORT.SELECT/= <17:15>
:958
:959     IDC=7
    
```

:960
:961 PORT.FUNC/= <14:13>
:962
:963 READ=0
:964 WRITE=1
:965 CONTROL=2
:966
:967 R.W.FUNC/= <12:10>
:968
:969 CSR=0
:970 DAR=1
:971 DATA.BYTE=2
:972 DATA.WORD=3
:973 PATTERN=4
:974 POSITION=5
:975
:976 CNTL.FUNC/= <12:10>
:977
:978 CLEAR.FIFO.CNTR=0
:979 RESET.BR=1
:980 CLEAR.IDC=3
:981 SET.AUTO=4
:982 CLEAR.AUTO=5
:983 SEL.FIFO.A=6
:984 SEL.FIFO.B=7
:985

```

:986 :PAGE
:987 : MEMORY REQUEST FIELD
:988 :
:989 : THIS FIELD IS USED TO INDICATE THE DESIRED OPERATION TO THE MEMORY CONTROL.
:990 : NOTE THAT THIS IS A DUMMY FIELD. IT IS ACTUALLY MADE UP OF TWO FIELDS.
:991 : M.FUNC1 AND M.FUNC2
:992 :
:993 : THE FOLLOWING SPECIAL CONSIDERATIONS MUST BE MADE:
:994 :
:995 : R. ATE.BYTE.RIGHT - PERFORMS 9-BIT ROTATE AND THEREFORE THE
:996 : ANSWER MUST BE CLEANED UP WITH A ROL WR[ ].
:997 :
:998 : WORD.SWAP - PERFORMS A 15-BIT ROTATE AND THEREFORE
:999 : THE ANSWER MUST BE CLEANED UP WITH A ROL WR[ ].
:1000 :
:1001 : OCTA.WRITE.P - MUST BE LONGWORD ALLIGNED.
:1002 :
:1003 : OCTA.WR.V.WCHK - MUST BE LONGWORD ALLIGNED. THIS DATA CAN
:1004 : CROSS PAGE BOUNDARIES
:1005 :
:1006 MEM.FUNC/=<7>,.VALIDITY=0
:1007 :
:1008 : PREFETCH=0 : USED TO TAKE VAR AND ADD ONE BEFORE REFERENCE.
:1009 : WRITE.TB=01 : WRITE TRANSLATION BUFFER
:1010 : TEST.V.RCHK=2 : TEST WITH VIRTUAL ADDRESS AND READ ACCESS CHECK
:1011 : READ.P=3 : READ WITH PHYSICAL ADDRESS
:1012 : WRITE.P=4 : WRITE WITH PHYSICAL ADDRESS
:1013 : TEST.V.WCHK=5 : WRITE WITH VIRTUAL ADDRESS AND WRITE ACCESS CHECK
:1014 : ROTATE.BYTE.RIGHT=6 : ROTATE ADDRESS DATA 9 BITS RIGHT AND RETURN
:1015 : WORD.SWAP=7 : ROTATE ADDRESS DATA 15 BITS LEFT AND RETURN
:1016 : READ.V.NOCHK=8 : READ WITH VIRTUAL ADDRESS AND READ ACCESS CHECK
:1017 : READ.V.WCHK=9 : READ WITH VIRTUAL ADDRESS AND WRITE ACCESS CHECK
:1018 : READ.V.RCHK=0A : READ WITH VIRTUAL ADDRESS AND READ ACCESS CHECK
:1019 : READ.MAINT.VAR.INC=0B : TEST VAR INCREMENTING.
:1020 : WRITE.V.NOCHK=0C : WRITE WITH VIRTUAL ADDRESS AND NO ACCESS CHECK
:1021 : WRITE.V.WCHK=0D : WRITE WITH VIRTUAL ADDRESS AND WRITE ACCESS CHECK
:1022 : IB.FILL=0E : READ WITH VIRTUAL ADDRESS AND READ ACCESS CHECK AND FILL IB
:1023 : READ.V.RCHK.IFILL=0F : READ WITH VIRTUAL ADDRESS AND READ ACCESS CHECK AND FILL IB
:1024 : WRITE.UBS.MAP=0F : WRITE TO THE UNIBUS MAP.
:1025 :
:1026 : OCTA.WRITE.P=10 : WRITE OCTA DATA WITH PHYSICAL ADDRESS.
:1027 : WRITE.TB.STEP=11 : WRITE TO TWO TB ENTRIES(WITH CORRECT ADDRESS)
:1028 : READ.UBS.MAP=12 : READ UNIBUS MAP
:1029 : READ.TB=13 : READ TRANSLATION BUFFER
:1030 : READ.MAINT.ADRS=14 : RPUTS VA ON UNIBUS LINES AND READS THIS DATA
:1031 : MAINT.ECC.DATA=15 : TEST ALL ECC FUNCTIONS.
:1032 : READ.CSR=16 : READ CONTROL REGISTER
:1033 : READ.CNTRL.REG=16 : READ CONTROL REGISTER
:1034 : WRITE.CNTRL.REG=17 : WRITE CONTROL REGISTER
:1035 : WRITE.CSR=17 : WRITE CONTROL REGISTER
:1036 : OCTA.READ.P=18 : READ OCTA DATA WITH PHYSICAL ADDRESS.
:1037 : READ.V.WCHK.LOCKED=19 : READ VIRTUAL ADDRESS AND WRITE ACCESS CHECK AND LOCKED
:1038 : OCTA.READ.V.RCHK=1A : READ OCTA WITH VIRTUAL ADDRESS AND READ ACCESS CHECK.
:1039 : READ.MAINT.BR.CHECK=1B : TESTS BRANCHING FUNCTION.
:1040 : ISSUE.BG=1C : ISSUES BUS GRANT (4+ADDRESS<1:0>)

```

:1041 OCTA.WR.V.WCHK=1D ; WRITE OCTA WITH VIRTUAL ADDRESS AND WRITE ACCESS CHECK.
 :1042 QUAD.READ.V.RCHK=1E ; READ QUAD WITH VIRTUAL ADDRESS AND READ ACCESS CHECK.
 :1043 READ.MAINT.UBS=1F ; PUTS VA ON UNIBUS LINES AND READS THIS AS DATA.
 :1044
 :1045
 :1046 M.FUNC2/=<18:16>,.VALIDITY=<DT.VAL>
 :1047
 :1048 M.FUNC1/=<8:7>,.VALIDITY=<DT.VAL>
 :1049
 :1050 PAR/=<23>,.DEFAULT=<.PARITY[<OPC2/>,<OS/>,<JUMP.ADRS/>,<JCTL/>]>
 :1051 .UCODE

```

:1052 .PAGE 'MACRO DEFINITIONS VER 00.02"
:1053
:1054 THE FORMAT FOR THIS GROUP IS TWO ADDRESS WITH A MODIFY DESTINATION.
:1055 THE FIRST OPERAND IS COMBINED WITH THE SECOND OPERAND AND THE
:1056 RESULT WRITTEN TO THE SECOND OPERAND. CMP AND BIT INSTRUCTIONS ARE
:1057 READ ONLY.
:1058
:1059 DURING ALL ARITHMETIC AND LOGICAL INSTRUCTIONS THE ALU CC'S
:1060 CAN BE LOADED BY ADDING THE
:1061
:1062 DT(SIZE)&SET.ALU.CC
:1063
:1064 MACRO TO THE MICROWORD. THIS SPECIFIES THAT ALU CC'S ARE LOADED
:1065 WITH THE 2901 CONDITIONS. NO EXTERNAL MUNGING IS DONE. IF THE
:1066 DATA OPERATION IS WITH BYTES THEN THE CC'S ARE SET
:1067 ACCORDING TO BITS 7-0. WORD USES BITS 15-0 AND LONGWORD USES BITS
:1068 31-0. THE FOLLOWING IS A DESCRIPTION OF THE ALU CC'S VALUE
:1069 FOR EACH FUNCTION:
:1070
:1071 ADD -- BINARY ADDITION OF THE TWO OPERANDS
:1072
:1073 N - SIGN BIT
:1074 Z - SET IF ANSWER IS ZERO.
:1075 V - ARITHMETIC OVERFLOW
:1076 C - CARRY
:1077
:1078 SUB -- BINARY ADDITION OF FIRST OPERAND AND TWO'S COMPLEMENT OF THE SECOND OPERAND.
:1079
:1080 N - SIGN BIT
:1081 Z - SET IF ANSWER IS ZERO.
:1082 V - ARITHMETIC OVERFLOW
:1083 C - COMPLEMENT OF THE BOPF ..
:1084
:1085 BIS -- LOGICAL OR OF THE TWO OPERANDS.
:1086
:1087 N - SIGN BIT
:1088 Z - SET IF ANSWER IS ZERO
:1089 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1090 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1091
:1092 BIC -- ONE'S COMPLEMENT OF FIRST OPERAND ANDED WITH SECOND OPERAND.
:1093
:1094 N - SIGN BIT
:1095 Z - SET IF ANSWER IS ZERO
:1096 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1097 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1098
:1099 AND -- LOGICAL AND OF THE TWO OPERANDS.
:1100
:1101 N - SIGN BIT
:1102 Z - SET IF ANSWER IS ZERO
:1103 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1104 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1105
:1106 XOR -- LOGICAL EXCLUSIVE OR OF THE TWO OPERANDS.

```



```

:1107 :
:1108 :
:1109 :
:1110 :
:1111 :
:1112 :
:1113 :
:1114 :
:1115 :
:1116 :
:1117 :
:1118 :
:1119 :
:1120 :
:1121 :
:1122 :
:1123 :
:1124 :
:1125 :
:1126 :
:1127 :
:1128 :
:1129 :
:1130 :
:1131 :
:1132 :

```

N - SIGN BIT
 Z - SET IF ANSWER IS ZERO
 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)

CMP -- PERFORM BINARY ADDITION OF FIRST OPERAND AND TWO'S COMPLEMENT OF SECOND OPERAND BUT DO NOT STORE.

N - SIGN BIT
 Z - SET IF ANSWER IS ZERO.
 V - ARITHMETIC OVERFLOW
 C - CARRY

BIT -- PERFORM LOGICAL AND OF THE TWO OPERANDS BUT DO NOT STORE.

N - SIGN BIT
 Z - SET IF ANSWER IS ZERO
 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)

SWAP -- SWITCH THE TWO VALUES.

N - SIGN BIT OF VALUE TO WR.
 Z - SET IF ANSWER IS ZERO OF VALUE TO WR.
 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)

```

:1133 .PAGE
:1134
:1135     MACROS FOR THE FORM OF WR[B]<-- WR[B] OPERATE LS[D]
:1136
:1137 ADD  LS[] TO  WR[]      'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP.<SHIFT[<.AND[<SDP/LS.PLUS.WR>,07F]>,-2]>'
:1138 SUB  LS[] FROM WR[]    'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP.<SHIFT[<.AND[<SDP/LS.FROM.WR>,07F]>,-2]>'
:1139 BIS  LS[] TO  WR[]    'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<SHIFT[<.AND[<SDP/LS.OR.WR>,OFF]>,-2]>'
:1140 BIC  LS[] TO  WR[]    'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<SHIFT[<.AND[<SDP/LS.MASK.WR>,OFF]>,-2]>'
:1141 AND  LS[] TO  WR[]    'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP.<SHIFT[<.AND[<SDP/LS.AND.WR>,07F]>,-2]>'
:1142 XOR  LS[] TO  WR[]    'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP.<SHIFT[<.AND[<SDP/LS.XOR.WR>,07F]>,-2]>'
:1143
:1144 CMP  LS[] WITH WR[]    'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<SHIFT[<.AND[<SDP/LS.CMP.WR>,OFF]>,-2]>'
:1145 BIT  LS[] WITH WR[]    'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<SHIFT[<.AND[<SDP/WR.BIT.LS>,OFF]>,-2]>'
:1146 SWAP LS[] WITH WR[]    'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<SHIFT[<.AND[<SDP/SWAP.LS.WR>,OFF]>,-2]>'
:1147
:1148     THE FOLLOWING MACROS IS TO BE USED WITH ADD,SUB,
:1149     AND,XOR LS[] TO WR[]
:1150
:1151     IT WILL TAKE THE ORIGINAL CONTENTS OF WR[]
:1152     AND PUT IT IN LS[].
:1153
:1154 XCHG      'XCHG.BIT/YES'
:1155
:1156     MACROS FOR THE FORM OF LS[D]<-- LS[D] OPERATE Q-REGISTER
:1157
:1158 ADD  Q TO  LS[]      'OPC/BASIC,D.ADRS/@1,DP/<SHIFT[<.AND[<SDP/Q.PLUS.LS>,OFF]>,-2]>'
:1159 SUB  Q FROM LS[]    'OPC/BASIC,D.ADRS/@1,DP/<SHIFT[<.AND[<SDP/Q.FROM.LS>,OFF]>,-2]>'
:1160 BIS  Q TO  LS[]    'OPC/BASIC,D.ADRS/@1,DP/<SHIFT[<.AND[<SDP/Q.OR.LS>,OFF]>,-2]>'
:1161 AND  Q TO  LS[]    'OPC/BASIC,D.ADRS/@1,DP/<SHIFT[<.AND[<SDP/Q.AND.LS>,OFF]>,-2]>'
:1162 XOR  Q TO  LS[]    'OPC/BASIC,D.ADRS/@1,DP/<SHIFT[<.AND[<SDP/Q.XOR.LS>,OFF]>,-2]>'
:1163
:1164 CMP  Q WITH LS[]    'OPC/BASIC,D.ADRS/@1,DP/<SHIFT[<.AND[<SDP/Q.CMP.LS>,OFF]>,-2]>'
:1165 BIT  Q WITH LS[]    'OPC/BASIC,D.ADRS/@1,DP/<SHIFT[<.AND[<SDP/LS.BIT.Q>,OFF]>,-2]>'
:1166
:1167     MACROS FOR THE FORM OF Q-REGISTER<-- Q-REGISTER OPERATE LS[D]
:1168
:1169 ADD  LS[] TO  Q      'OPC/BASIC,D.ADRS/@1,DP/<SHIFT[<.AND[<SDP/LS.PLUS.Q>,OFF]>,-2]>'
:1170 SUB  LS[] FROM Q    'OPC/BASIC,D.ADRS/@1,DP/<SHIFT[<.AND[<SDP/LS.FROM.Q>,OFF]>,-2]>'
:1171 BIS  LS[] TO  Q      'OPC/BASIC,D.ADRS/@1,DP/<SHIFT[<.AND[<SDP/LS.OR.Q>,OFF]>,-2]>'
:1172 BIC  LS[] TO  Q      'OPC/BASIC,D.ADRS/@1,DP/<SHIFT[<.AND[<SDP/LS.MASK.Q>,OFF]>,-2]>'
:1173 AND  LS[] TO  Q      'OPC/BASIC,D.ADRS/@1,DP/<SHIFT[<.AND[<SDP/LS.AND.Q>,OFF]>,-2]>'
:1174 XOR  LS[] TO  Q      'OPC/BASIC,D.ADRS/@1,DP/<SHIFT[<.AND[<SDP/LS.XOR.Q>,OFF]>,-2]>'
:1175
:1176 CMP  LS[] WITH Q    'OPC/BASIC,D.ADRS/@1,DP/<SHIFT[<.AND[<SDP/LS.CMP.Q>,OFF]>,-2]>'
:1177 BIT  LS[] WITH Q    'OPC/BASIC,D.ADRS/@1,DP/<SHIFT[<.AND[<SDP/LS.BIT.Q>,OFF]>,-2]>'
:1178
:1179     MACROS FOR THE FORM OF LS[D]<-- LS[D] OPERATE WR[B]
:1180
:1181 ADD  WR[] TO  LS[]    'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<SHIFT[<.AND[<SDP/WR.PLUS.LS>,OFF]>,-2]>'
:1182 SUB  WR[] FROM LS[]  'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<SHIFT[<.AND[<SDP/WR.FROM.LS>,OFF]>,-2]>'
:1183 BIS  WR[] TO  LS[]    'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<SHIFT[<.AND[<SDP/WR.OR.LS>,OFF]>,-2]>'
:1184 AND  WR[] TO  LS[]    'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<SHIFT[<.AND[<SDP/WR.AND.LS>,OFF]>,-2]>'
:1185 XOR  WR[] TO  LS[]    'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<SHIFT[<.AND[<SDP/WR.XOR.LS>,OFF]>,-2]>'
:1186
:1187 CMP  WR[] WITH LS[]    'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<SHIFT[<.AND[<SDP/WR.CMP.LS>,OFF]>,-2]>'
    
```

```

:1188 BIT WR[] WITH LS[]      'OPC1/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.BIT.LS>,0FF]>,-2]>'
:1189 SWAP WR[] WITH LS[]    'SWAP LS[@2] WITH WR[@1]'
:1190
:1191      MACROS FOR THE FORM OF WR[B]<-- WR[B] OPERATE WR[A]
:1192
:1193 ADD WR[] TO WR[]          'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.PLUS.WR>,03F]>'
:1194 SUB WR[] FROM WR[]       'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.FROM.WR>,03F]>'
:1195 BIS WR[] TO WR[]         'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.OR.WR>,03F]>'
:1196 BIC WR[] TO WR[]         'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.MASK.WR>,03F]>'
:1197 AND WR[] TO WR[]         'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.AND.WR>,03F]>'
:1198 XOR WR[] TO WR[]         'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.XOR.WR>,03F]>'
:1199
:1200 CMP WR[] WITH WR[]       'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.CMP.WR>,03F]>'
:1201 BIT WR[] WITH WR[]      'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.BIT.WR>,03F]>'
:1202
:1203      MACROS FOR THE FORM OF WR[B]<-- WR[B] OPERATE Q-REGISTER
:1204
:1205 ADD Q TO WR[]             'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.PLUS.WR>,03F]>'
:1206 SUB Q FROM WR[]          'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.FROM.WR>,03F]>'
:1207 BIS Q TO WR[]            'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.OR.WR>,03F]>'
:1208 AND Q TO WR[]            'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.AND.WR>,03F]>'
:1209 XOR Q TO WR[]            'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.XOR.WR>,03F]>'
:1210
:1211 CMP Q WITH WR[]          'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.CMP.WR>,03F]>'
:1212 BIT Q WITH WR[]         'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.BIT.WR>,03F]>'
:1213
:1214      MACROS FOR THE FORM OF Q-REGISTER<-- Q-REGISTER OPERATE WR[B]
:1215
:1216 ADD WR[] TO Q             'OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.PLUS.Q>,03F]>'
:1217 SUB WR[] FROM Q           'OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.FROM.Q>,03F]>'
:1218 BIS WR[] TO Q            'OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.OR.Q>,03F]>'
:1219 BIC WR[] TO Q            'OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.MASK.Q>,03F]>'
:1220 AND WR[] TO Q            'OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.AND.Q>,03F]>'
:1221 XOR WR[] TO Q            'OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.XOR.Q>,03F]>'
:1222
:1223 CMP WR[] WITH Q          'OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.CMP.Q>,03F]>'
:1224 BIT WR[] WITH Q         'OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/Q.BIT.WR>,03F]>'

```

```

:1225 .PAGE
:1226
:1227 THE FORMAT OF THIS GROUP IS THREE ADDRESS TYPE. THE FIRST AND SECOND
:1228 OPERANDS ARE READ AND THE RESULT IS WRITTEN INTO THE THIRD OPERAND.
:1229
:1230 MACROS FOR THE FORM OF Q-REGISTER<-- WR[B] OPERATE WR[A]
:1231
:1232 ADD WR[] PLUS WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.PLUS.WR.Q>,03F]>''
:1233 SUB WR[] FROM WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.FROM.WR.Q>,03F]>''
:1234 BIS WR[] WITH WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.OR.WR.Q>,03F]>''
:1235 BIC WR[] WITH WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.MASK.WR.Q>,03F]>''
:1236 AND WR[] WITH WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.AND.WR.Q>,03F]>''
:1237 XOR WR[] WITH WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.XOR.WR.Q>,03F]>''
:1238
:1239 MACROS FOR THE FORM OF Q-REGISTER<-- WR[B] OPERATE LS[D]
:1240
:1241 ADD WR[] PLUS LS[] TO Q 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.PLUS.LS.Q>,OFF]>,-2]>''
:1242 SUB WR[] FROM LS[] TO Q 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.FROM.LS.Q>,OFF]>,-2]>''
:1243 BIS WR[] WITH LS[] TO Q 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.OR.LS.Q>,OFF]>,-2]>''
:1244 AND WR[] WITH LS[] TO Q 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.AND.LS.Q>,OFF]>,-2]>''
:1245 XOR WR[] WITH LS[] TO Q 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.XOR.LS.Q>,OFF]>,-2]>''
:1246
:1247 MACROS FOR THE FORM OF Q-REGISTER <-- LS[D] OPERATE WR[B]
:1248
:1249 ADD LS[] PLUS WR[] TO Q 'ADD WR[@2] PLUS LS[@1] TO Q'
:1250 SUB LS[] FROM WR[] TO Q 'OPC/BASIC,B.ADRS/@2,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/LS.FROM.WR.Q>,OFF]>,-2]>''
:1251 BIS LS[] WITH WR[] TO Q 'BIS WR[@2] WITH LS[@1] TO Q'
:1252 AND LS[] WITH WR[] TO Q 'AND WR[@2] WITH LS[@1] TO Q'
:1253 XOR LS[] WITH WR[] TO Q 'XOR WR[@2] WITH LS[@1] TO Q'
:1254
:1255

```

```

:1256 .PAGE
:1257 ; SPECIAL FUNCTION ADD/SUB OPERATIONS
:1258
:1259 ADD (WR[] TO WR[])+1      'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.PLUS.WR+1>,03F]>'
:1260 ADD (LS[] TO WR[])+1      'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.PLUS.WR+1>,OFF]>,-2]>'
:1261 ADD (WR[] TO LS[])+1      'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.PLUS.LS+1>,OFF]>,-2]>'
:1262
:1263 ADD OS PLUS WR[] TO LS[]   'OPC1/MOVE,B.ADRS/@1,XD.ADRS/@2,MDP/<.SHIFT[<.AND[<SDP/WR.PLUS.OS>,3F]>,-3]>'
:1264
:1265 SUB (WR[] FROM WR[])-1      'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.FROM.WR-1>,03F]>'
:1266 SUB (LS[] FROM WR[])-1      'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.FROM.WR-1>,OFF]>,-2]>'
:1267 SUB (WR[] FROM LS[])-1      'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.FROM.LS-1>,OFF]>,-2]>'
:1268
:1269
:1270 SUB WRB[] FROM WRA[] TO WRB[] 'OPC1/EXTENDED,B.ADRS/@1,A.ADRS/@2,XDP/<.AND[<SDP/WR.FROM.WR.WR>,03F]>'
:1271 SUB LS[] FROM WR[] TO LS      'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.FROM.WR.LS>,OFF]>,-2]>'
:1272 SUB WR[] FROM LS[] TO WR      'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WRA.FROM.LS.WRB>,OFF]>,-2]>'
:1273
:1274 SUB WRA[] FROM Q TO WRB[]      'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.FROM.Q.WR>,03F]>'
:1275 SUB LS[] FROM Q TO LS[]      'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/LS.FROM.Q.LS>,OFF]>,-2]>'
:1276 SUB Q FROM WR[] TO Q          'OPC1/EXTENDED,B.ADRS/@1,XDP/<.AND[<SDP/Q.FROM.WR.Q>,03F]>'
:1277 SUB Q FROM LS[] TO Q          'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/Q.FROM.LS.Q>,OFF]>,-2]>'
:1278
:1279 ADD Q PLUS WR[] TO LS[]      'OPC/BASIC,D.ADRS/@2,B.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/Q.PLUS.WR.TO.LS>,OFF]>,-2]>'
:1280 SUB Q FROM WR[] TO LS[]      'OPC/BASIC,D.ADRS/@2,B.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/Q.FROM.WR.TO.LS>,OFF]>,-2]>'
:1281 ADD Q PLUS LS[] TO WR[]      'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/Q.PLUS.LS.TO.WR>,OFF]>,-2]>'
:1282
:1283 ADD Q TO WR[] XCHG TO LS[]    'OPC/BASIC,D.ADRS/@2,B.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/WR.PLUS.Q.XCHG>,OFF]>,-2]>'
:1284

```

```

1285 .PAGE
1286 : MOVE MACRO INSTRUCTIONS
1287 :
1288 : DURING THE MOVE INSTRUCTIONS THE ALU CC'S CAN BE LOADED BY
1289 : ADDING THE
1290 :
1291 : DT(SIZE)&SET.ALU.CC
1292 :
1293 : MACRO TO THE MICROWORD. THE FOLLOWING IS A DESCRIPTION OF THE ALU
1294 : CC'S VALUE FOR EACH FUNCTION:
1295 :
1296 : MOV -- PUT FIRST OPERAND ON TOP OF THE SECOND OPERAND.
1297 :
1298 : N - SIGN BIT
1299 : Z - SET IF ANSWER IS ZERO
1300 : V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
1301 : C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
1302 :
1303 : MCOM -- PUT ONE'S COMPLEMENT OF THE FIRST OPERAND ON TOP OF THE SECOND OPERAND.
1304 :
1305 : N - SIGN BIT
1306 : Z - SET IF ANSWER IS ZERO
1307 : V - RANJOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
1308 : C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
1309 :
1310 : MNEG -- PUT TWO'S COMPLEMENT OF THE FIRST OPERAND ON TOP OF THE SECOND OPERAND.
1311 :
1312 : N - SIGN BIT
1313 : Z - SET IF ANSWER IS ZERO.
1314 : V - ARITHMETIC OVERFLOW
1315 : C - CARRY
1316 : MOV Q TO LS[] 'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/MOV.Q.LS>,OFF]>,-2]>'
1317 : MOV Q TO WR[] 'OPC1/EXTENDED,B.ADRS/@1,XDP/<.AND[<SDP/MOV.Q.WR>,03F]>'
1318 : MOV LS[] TO Q 'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/MOV.LS.Q>,OFF]>,-2]>'
1319 : MOV Q TO WR[] XCHG TO LS[] 'OPC/BASIC,D.ADRS/@2,B.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/MOV.Q.WR&WR.LS>,OFF]>,-2]>'
1320 :
1321 : MOV IB.DATA TO OS 'OPC2/DECODE,IFUNC/SPEC,R.DST/NOP,IB.REQ/IB.REQ,JCTL/NO.JUMP,TST,LD.OS/LOAD.OS'
1322 : MOV CM.IB.DATA TO OS 'OPC2/DECODE,IFUNC/SPEC,R.DST/NOP,IB.REQ/NOP,JCTL/NO.JUMP,TST,LD.OS/LOAD.OS'
1323 :
1324 : MOV MEM.DATA TO WR[] 'OPC1/MOVE,B.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.MEM.WR>,3F]>,-3]>,XD.ADRS/WR.BACKUP'
1325 : MOV MEM.DATA TO WR[] XCHG TO LS[] 'OPC1/MOVE,B.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.MEM.WR>,3F]>,-3]>,XD.ADRS/@2'
1326 : MOV MEM.DATA TO LS[] 'OPC1/MOVE,XD.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.MEM.LS>,3F]>,-3]>'
1327 : MOV LS[] TO WR[] 'OPC1/MOVE,XD.ADRS/@1,B.ADRS/@2,MDP/<.SHIFT[<.AND[<SDP/MOV.LS.WR>,3F]>,-3]>'
1328 : MOV WR[] TO LS[] 'OPC1/MOVE,B.ADRS/@1,XD.ADRS/@2,MDP/<.SHIFT[<.AND[<SDP/MOV.WR.LS>,3F]>,-3]>'
1329 : MOV WR[] TO WR[] 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.PLUS.0.TO.WR>,03F]>'
1330 : MOV WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.OR.WR.Q>,03F]>'
1331 : MOV XWR[] TO Q 'OPC1/MOVE,XB.ADRS/@1,XD.ADRS/0,MDP/<.SHIFT[<.AND[<SDP/MOV.XWR.Q>,3F]>,-3]>'
1332 :
1333 : MOV FPA TO LS[] 'OPC1/MOVE,XD.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.ACC.LS>,3F]>,-3]>'
1334 : MOV PORT TO LS[] 'OPC1/MOVE,XD.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.ACC.LS>,3F]>,-3]>,CC/DT(LONG)&HOLD.ALU.CC'
1335 :
1336 : MCOM WR[] TO LS[] 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.COM.LS>,OFF]>,-2]>'
1337 : MCOM LS[] TO WR[] 'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.COM.WR>,OFF]>,-2]>'
1338 : MNEG WR[] TO LS[] 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.NEG.LS>,OFF]>,-2]>'
1339 : MNEG LS[] TO WR[] 'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.NEG.WR>,OFF]>,-2]>'

```

:1340	MNEG	WR[]	TO	WR[]	'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.NEG.WR>,03F]>''
:1341	MCOM	WR[]	TO	WR[]	'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.COM.WR>,03F]>''
:1342	MINC	WR[]	TO	WR[]	'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.INC.WR>,03F]>''
:1343	MDEC	WR[]	TO	WR[]	'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.DEC.WR>,03F]>''
:1344	MNEG	Q	TO	LS[]	'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/Q.NEG.LS>,OFF]>,-2]>''

```

:1345 .PAGE
:1346
:1347 SINGLE OPERAND OPERATIONS
:1348
:1349 THIS FORM OF INSTRUCTION PERFORMS THE DESIRED OPERATION ON THE OPERAND
:1350 SPECIFIED.
:1351
:1352 CLEAR
:1353 NEGATE
:1354 INCREMENT
:1355 DECREMENT
:1356 COMPLEMENT
:1357 TEST
:1358
:1359 DURING THE SINGLE OPERAND INSTRUCTIONS THE ALU CC'S CAN BE
:1360 LOADING USING THE
:1361
:1362 DT(SIZE)&SET.ALU.CC
:1363
:1364 MACRO IN THE MICROWORD. THE FOLLOWING IS A DESCRIPTION OF THE
:1365 ALU CC'S FOR EACH FUNCTION:
:1366
:1367 CLR -- STORE A ZERO IN THE OPERAND.
:1368
:1369 N - SIGN BIT
:1370 Z - SET IF ANSWER IS ZERO
:1371 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1372 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1373
:1374 NEG -- PERFORM TWO'S COMPLEMENT OF THE OPERAND.
:1375
:1376 N - SIGN BIT
:1377 Z - SET IF ANSWER IS ZERO.
:1378 V - ARITHMETIC OVERFLOW
:1379 C - CARRY
:1380
:1381 INC -- ADD ONE TO THE OPERAND.
:1382
:1383 N - SIGN BIT
:1384 Z - SET IF ANSWER IS ZERO.
:1385 V - ARITHMETIC OVERFLOW
:1386 C - CARRY
:1387
:1388 DEC -- SUBTRACT ONE FROM THE OPERAND.
:1389
:1390 N - SIGN BIT
:1391 Z - SET IF ANSWER IS ZERO.
:1392 V - ARITHMETIC OVERFLOW
:1393 C - CARRY
:1394
:1395 COM -- PERFORM ONE'S COMPLEMENT OF THE OPERAND (XNOR WITH ZERO).
:1396
:1397 N - SIGN BIT
:1398 Z - SET IF ANSWER IS ZERO.
:1399 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)

```



```

:1400 : C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1401 :
:1402 : TST -- ADD ZERO TO THE OPERAND TO SET THE CONDITION CODES.
:1403 :
:1404 : N - SIGN BIT
:1405 : Z - SET IF ANSWER IS ZERO
:1406 : V - ARITHMETIC OVERFLOW (IN THIS CASE ALWAYS ZERO)
:1407 : C - CARRY (IN THIS CASE ZERO)
:1408 :
:1409 :
:1410 CLR Q 'OPC1/EXTENDED,A.ADRS/0,B.ADRS/0,XDP/<.AND[<SDP/WR.XOR.WR.Q>,03F]>'
:1411 CLR LS[] 'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/CLR.LS>,OFF]>,-2]>'
:1412 CLR WR[] 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.XOR.WR>,03F]>'
:1413 :
:1414 NEG Q 'OPC1/EXTENDED,XDP/<.AND[<SDP/NEG.Q>,03F]>'
:1415 NEG LS[] 'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/NEG.LS>,OFF]>,-2]>'
:1416 NEG WR[] 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.NEG.WR>,03F]>'
:1417 :
:1418 INC Q 'OPC1/EXTENDED,XDP/<.AND[<SDP/INC.Q>,03F]>'
:1419 INC LS[] 'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/INC.LS>,OFF]>,-2]>'
:1420 INC WR[] 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.INC.WR>,03F]>'
:1421 :
:1422 DEC Q 'OPC1/EXTENDED,XDP/<.AND[<SDP/DEC.Q>,03F]>'
:1423 DEC LS[] 'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/DEC.LS>,OFF]>,-2]>'
:1424 DEC WR[] 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.DEC.WR>,03F]>'
:1425 :
:1426 COM Q 'OPC1/EXTENDED,XDP/<.AND[<SDP/COM.Q>,03F]>'
:1427 COM LS[] 'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/COM.LS>,OFF]>,-2]>'
:1428 COM WR[] 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.COM.WR>,03F]>'
:1429 :
:1430 TST WR[] 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.PLUS.0.TO.WR>,03F]>'
:1431 TST Q 'OPC1/EXTENDED,XDP/<.AND[<SDP/Q.PLUS.0>,03F]>'
:1432 :
:1433 NOP 'OPC/BASIC,D.ADRS/0,B.ADRS/0,DP/<.SHIFT[<.AND[<SDP/Q.CMP.LS>,OFF]>,-2]>'

```

:1434 :PAGE

:1435 :
:1436 :MACROS FOR SHIFT OPERATION ON WR[B] AND/OR Q-REGISTER

:1437 :
:1438 :DURING THE SHIFT OPERATIONS THE ALU CC'S CAN BE LOADED BY
:1439 :USING THE

:1440 :
:1441 :DT(SIZE)&SET.ALU.CC

:1442 :
:1443 :MACRO WITH THE MICROWORD. THE FOLLOWING IS A DESCRIPTION OF THE
:1444 :ALU CC'S WITH EACH FUNCTION.

:1445 :
:1446 :ROR WR[] -- ROTATE 32 BIT VALUE ONE POSITION RIGHT.

:1447 :
:1448 :N - SIGN OF INPUT
:1449 :Z - SET IF INPUT IS ZERO
:1450 :V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1451 :C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)

:1452 :
:1453 :ROL WR[] -- ROTATE 32 BIT VALUE ONE POSITION LEFT.

:1454 :
:1455 :N - SIGN OF INPUT
:1456 :Z - SET IF INPUT IS ZERO
:1457 :V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1458 :C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)

:1459 :
:1460 :ASHR WR[] -- SHIFT 32 BIT VALUE RIGHT ONE POSITION AND SIGN EXTEND.

:1461 :
:1462 :N - SIGN OF INPUT
:1463 :Z - SET IF INPUT IS ZERO
:1464 :V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1465 :C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)

:1466 :
:1467 :ASHL WR[] -- SHIFT 32 BIT VALUE LEFT ONE POSITION AND INSERT A ZERO IN BOTTOM BIT.

:1468 :
:1469 :N - SIGN OF INPUT
:1470 :Z - SET IF INPUT IS ZERO
:1471 :V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1472 :C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)

:1473 :
:1474 :ASHRC WR[].Q -- SHIFT 64 BIT WR.Q ONE POSITION LEFT AND SIGN EXTEND.

:1475 :
:1476 :N - SIGN OF INPUT WR[]
:1477 :Z - SET IF INPUT WR[] IS ZERO
:1478 :V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1479 :C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)

:1480 :
:1481 :RORC WR[].Q -- ROTATE 64 BIT WR.Q ONE POSITION RIGHT.

:1482 :
:1483 :N - SIGN OF INPUT WR[]
:1484 :Z - SET IF INPUT WR[] IS ZERO
:1485 :V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1486 :C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)

:1487 :
:1488 :ASHLC WR[].Q -- SHIFT 64 BIT WR.Q ONE POSITION LEFT AND INSERT ZERO.

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) MACRO DEFINITIONS
3-MAR-82 10:02:46
MACRO DEFINITIONS

C 4
3-MAR-82

Fiche 1 Frame C4

Sequence 41

ENKCB Micro-diagnostic for DAP module Rev 01.00

Page 35

```
:1489 :
:1490 :
:1491 : N - SIGN OF INPUT WR[]
:1492 : Z - SET IF INPUT WR[] IS ZERO
:1493 : V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1494 : C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1495 :
:1496 : ROLC WR[].Q -- ROTATE 64 BIT WR.Q ONE POSITION LEFT.
:1497 :
:1498 : N - SIGN OF INPUT WR[]
:1499 : Z - SET IF INPUT WR[] IS ZERO
:1500 : V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1501 : C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1502 :
:1503 : SHL2 WR[] -- SHIFT 32 BIT VALUE TWO POSITIONS LEFT INSERTING ZEROES IN THE BOTTOM BITS.
:1504 :
:1505 : N - SIGN OF INPUT WR[]+WR[]
:1506 : Z - SET IF INPUT WR[]+WR[] IS ZERO
:1507 : V - OVERFLOW OF INPUT WR[]+WR[]
:1508 : C - CARRY OF INPUT WR[]+WR[]
:1509 : ROR WR[] "OPC1/EXTENDED,B.ADRS/@1,SI/ROT.WR,XDP/<.AND[<SDP/ASHR>,03F]>"
:1510 :
:1511 : ROL WR[] "OPC1/EXTENDED,B.ADRS/@1,SI/ROT.WR,XDP/<.AND[<SDP/ASHL>,03F]>"
:1512 :
:1513 :
:1514 : ASHR WR[] "OPC1/EXTENDED,B.ADRS/@1,SI/ASH.WR,XDP/<.AND[<SDP/ASHR>,03F]>"
:1515 :
:1516 : ASHL WR[] "OPC1/EXTENDED,B.ADRS/@1,SI/ASH.WR,XDP/<.AND[<SDP/ASHL>,03F]>"
:1517 :
:1518 : ASHRC WR[].Q "OPC1/EXTENDED,B.ADRS/@1,SI/ASH.WR.Q,XDP/<.AND[<SDP/ASHRC>,03F]>"
:1519 :
:1520 : RORC WR[].Q "OPC1/EXTENDED,B.ADRS/@1,SI/ROT.WR.Q,XDP/<.AND[<SDP/ASHRC>,03F]>"
:1521 :
:1522 : ASHLC WR[].Q "OPC1/EXTENDED,B.ADRS/@1,SI/ASH.WR.Q,XDP/<.AND[<SDP/ASHLC>,03F]>"
:1523 :
:1524 : ROLC WR[].Q "OPC1/EXTENDED,B.ADRS/@1,SI/ROT.WR.Q,XDP/<.AND[<SDP/ASHLC>,03F]>"
:1525 :
:1526 : SHL2 WR[] "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,SI/2,XDP/<.AND[<SDP/SHL2>,03F]>"
:1527 :
:1528 : COND.ADD&SHIFT WR[] TO WR[].Q "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,SI/MUL.SHF,XDP/<.AND[<SDP/COND.ADD&SHIFT>,03F]>"
:1529 :
:1530 : COND.SUB&SHIFT WR[] FROM WR[].Q "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,SI/MUL.SHF,XDP/<.AND[<SDP/COND.SUB&SHIFT>,03F]>"
:1531 :
:1532 : UNSIGNED.MUL WR[] TO WR[].Q "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,SI/UMUL.SHF,XDP/<.AND[<SDP/COND.ADD&SHIFT>,03F]>"
:1533 : SHR WR[] "OPC1/EXTENDED,B.ADRS/@1,SI/SHF.WR.Q,XDP/<.AND[<SDP/ASHR>,03F]>"
:1534 : SHL WR[] "ASHL WR[@1]"
:1535 : SHRC WR[].Q "OPC1/EXTENDED,B.ADRS/@1,SI/SHF.WR.Q,XDP/<.AND[<SDP/ASHRC>,03F]>"
:1536 : SHLC WR[].Q "ASHLC WR[@1]"
:1537 :
:1538 :
:1539 : : MEMORY REQUEST MACRO
:1540 :
:1541 : * THIS MACRO IS NOT FOR GENERAL USE!!! *
:1542 : * THIS IS ONLY FOR USE IN THE FOLLOWING MACRO *
:1543 : MEM.FUNC[] "M.FUNC2/<.SHIFT[<MEM.FUNC/@1>,-2]>,M.FUNC1/<.AND[<MEM.FUNC/@1>,3]>"
```

:1544
:1545 MEM.REQ[] ADRS[] DT[]
:1546 WRITE.MEM LS[]

''OPC2/MEM.REQ, MEM.FUNC[@1], D.ADRS/@2, MDT/@3''
''OPC1/MOVE, XD.ADRS/@1, MDP/<.SHIFT[<.AND[<SDP/MOV.LS.MEM>, 3F]>, -3]>''

```

:1547 .PAGE
:1548
:1549 MACROS FOR MISCELLANEOUS OPERATIONS
:1550
:1551 IN THE NEBULA MICROPROGRAMMING LANGUAGE THERE IS A NEED FOR
:1552 VARIOUS UNIQUE FUNCTIONAL OPERATIONS. THESE ARE FOUND IN THE
:1553 MISC INSTRUCTIONS. MOST FUNCTIONS ARE EXECUTED BY INSERTING
:1554 THE REQUEST INSIDE THE MISC[] AND MISC2[] MACROS.
:1555
:1556 MISC [] 'OPC2/MISC,MISC.1/@1,MISC.2/NOP,MISC.PORT/MISC'
:1557 MISC2 [] 'OPC2/MISC,MISC.1/NOP,MISC.2/@1,MISC.PORT/MISC'
:1558 MISC [] WR[] 'MISC [a1],MISC.WRL/0,MISC.WRR/@2'
:1559 MISC2 [] WR[] 'MISC2 [a1],MISC.WRL/0,MISC.WRR/@2'
:1560
:1561 PORT.CONTROL [] 'OPC2/MISC,MISC.PORT/PORT,CNTL.FUNC/@1,PORT.SELECT/IDC,PORT.FUNC/CONTROL'
:1562 PCRT.WRITE PORT.ADRS[] WITH WR[] 'OPC2/MISC,MISC.PORT/PORT,R.W.FUNC/@1,PORT.SELECT/IDC,PORT.FUNC/WRITE,MISC.WRL/0,MIS
:1563 PORT.READ PORT.ADRS[] 'OPC2/MISC,MISC.PORT/PORT,R.W.FUNC/@1,PORT.SELECT/IDC,PORT.FUNC/READ'
:1564
:1565 A FEW CASES OF THE MISCELLANEOUS FUNCTIONS HAVE BEEN CALLED OUT AS
:1566 UNIQUE FUNCTIONS IN THE MICROPROGRAMMING LANGUAGE SET. THESE
:1567 ARE LISTED BELOW.
:1568
:1569 THE FOLLOWING TO MACROS SEND DATA TO EXTERNAL ACCELERATORS.
:1570
:1571 ASSERT.DA.AND.PASS.WRO 'MISC2 [ASSERT.DA.AND.PASS.WR] WR[0]'
:1572 ASSERT.DA.AND.PASS.WR1 'MISC2 [ASSERT.DA.AND.PASS.WR] WR[1]'
:1573
:1574 STATE REGISTER MACROS - THE POSSIBLE CHANGES TO THE STATE
:1575 BITS ARE SPECIFIED AS INDIVIDUAL FUNCTIONS.
:1576
:1577 CLR.STATE.ZERO 'OPC2/MISC,MISC.1/CLR.S0,MISC.2/NOP,MISC.PORT/MISC'
:1578 CLR.STATE.ONE 'OPC2/MISC,MISC.1/CLR.S1,MISC.2/NOP,MISC.PORT/MISC'
:1579 SET.STATE.ZERO 'OPC2/MISC,MISC.1/SET.S0,MISC.2/NOP,MISC.PORT/MISC'
:1580 SET.STATE.ONE 'OPC2/MISC,MISC.1/SET.S1,MISC.2/NOP,MISC.PORT/MISC'
:1581 SET.STATE.ZERO&CLR.STATE.ONE 'OPC2/MISC,MISC.1/CLR.S1&SET.S0,MISC.2/NOP,MISC.PORT/MISC'
:1582 SET.STATE.ONE&CLR.STATE.ZERO 'OPC2/MISC,MISC.1/SET.S1&CLR.S0,MISC.2/NOP,MISC.PORT/MISC'
:1583 CLR.STATE.ONE&CLR.STATE.ZERO 'OPC2/MISC,MISC.1/CLR,MISC.2/NOP,MISC.PORT/MISC'
:1584 SET.BACKUP.MASK.VALID 'OPC2/MISC,MISC.1/SET.BACKUP.MASK.VALID,MISC.2/NOP,MISC.PORT/MISC'
:1585
:1586
:1587 ; CONTEXT SIZE AND CONDITION CODE MACROS
:1588
:1589 DT(SIZE)&MAP.ALU.CC.TO.PSL 'CC/DT(SIZE)&MAP.ALU.CC.TO.PSL'
:1590 DT(LONG)&SET.ALU.CC 'CC/DT(LONG)&SET.ALU.CC'
:1591 DT(SIZE)&SET.ALU.CC 'CC/DT(SIZE)&SET.ALU.CC'
:1592
  
```

```

:1593 .PAGE
:1594 JUMP AND MICRO SEQUENCER CONTROL MACROS
:1595
:1596 THE FOLLOWING MACROS ARE USED TO CONTROL THE PROGRAMS FLOW OF
:1597 CONTROL.
:1598
:1599 JMP [] -- TRANSFER TO NEW ADDRESS UNCONDITIONALLY.
:1600
:1601 JMP.OS [] -- TRANSFER TO RESULTANT ADDRESS OF LOGICAL OR OF ADDRESS
:1602 IN THE INSTRUCTION AND THE OS REGISTER UNCONDITIONALLY.
:1603
:1604 JMP.IF[] TO [] -- TRANSFER TO NEW ADDRESS ONLY IF JUMP
:1605 CONDITION IS MET.
:1606
:1607 JSR [] -- TRANSFER TO NEW ADDRESS UNCONDITIONALLY AND PUT
:1608 RETURN ADDRESS ON THE SEQUENCER STACK.
:1609
:1610 JSR.OS [] -- TRANSFER TO RESULTANT ADDRESS OF LOGICAL OR OF
:1611 ADDRESS IN THE INSTRUCTION AND THE OS REGISTER.
:1612 THE RETURN ADDRESS IS ALSO PUSHED ON THE SEQUENCER STACK.
:1613
:1614 RETURN -- TRANSFER TO ADDRESS ON TOP OF THE SEQUENCER STACK AND
:1615 POP THE STACK.
:1616
:1617 RETURN+1 -- TRANSFER TO ADDRESS ON TOP OF THE SEQUENCER
:1618 STACK PLUS ONE AND POP THE STACK.
:1619
:1620 RETURN+1.IF(MEM.REF.OK) -- IF THE MOST RECENT MEMORY REQUEST HAD
:1621 NO ERROR THEN TRANSFER TO ADDRESS ON TOP OF THE SEQUENCER
:1622 STACK PLUS ONE AND POP THE STACK. IF THERE WAS AN
:1623 ERROR THEN SKIP OVER THE ENEXT INSTRUCTION.
:1624
:1625 LOOP.IF(NEQ) -- IF THE ALU Z-BIT IS ZERO (LAST VALUE CALCULATED
:1626 WHEN THE ALU.CC'S WERE SET DID NOT RESULT AS ZERO)
:1627 THEN TRANSFER TO THE ADDRESS ON TOP OF THE SEQUENCER
:1628 STACK. THE STACK IS NOT POPPED. IF THE ALU Z-BIT
:1629 IS ONE THEN CONTINUE WITH THE NEXT INSTRUCTION.
:1630 (UPC+1) AND THE STACK IS POPPED.
:1631
:1632 LOOP.IF(GEQU) -- IF THE ALU C-BIT IS ONE (LAST VALUE CALCULATED WHEN
:1633 THE ALU.CC'S WERE SET WAS GREATER THAN OR EQUAL TO ZERO)
:1634 THEN TRANSFER TO THE ADDRESS ON TOP OF THE SEQUENCER
:1635 STACK. THE STACK IS NOT POPPED. IF THE ALU C-BIT
:1636 IS ZERO THEN CONTINUE WITH THE NEXT INSTRUCTION
:1637 (UPC+1) AND THE STACK IS POPPED.
:1638
:1639 LOOP.IF(NO INTERRUPT AND NO SYNC) -- IF THE INTERRUPT CONDITION
:1640 IS NOT TRUE AND THE ACCELERATOR SYNC CONDITION IS NOT
:1641 TRUE THEN TRANSFER TO THE ADDRESS ON THE TOP OF THE SEQUENCER
:1642 STACK. THE STACK IS NOT POPPED. IF THE INTERRUPT
:1643 CONDITION IS TRUE THEN CONTINUE WITH THE NEXT INSTRUCTION
:1644 (UPC+1) AND THE SEQUENCER STACK IS NOT POPPED. IF THE ACCELERATOR
:1645 SYNC IS TRUE THEN CONTINUE WITH THE NEXT INSTRUCTION
:1646 (UPC+1) AND THE SEQUENCER STACK IS POPPED.
:1647

```

```

:1648 :      SKIP.IF[] -- IF THE SKIP CONDITION IS SATISFIED THEN TRANSFER
:1649 :      TO UPC+2 ELSE TRANSFER TO UPC+1.
:1650 :
:1651 :
:1652 JMP []      "OPC2/JUMP,JUMP.ADRS/@1,JCTL/JUMP"
:1653 JMP.OS []   "OPC2/JUMP,JUMP.ADRS/@1,OS/WITH.OS,JCTL/JUMP"
:1654 JMP.IF[] TO [] "OPC2/JUMP,JCTL/@1,JUMP.ADRS/@2"
:1655 JSR []      "OPC2/JUMP,JCTL/JSR,JUMP.ADRS/@1"
:1656 JSR.OS []   "OPC2/JUMP,JCTL/JSR,JUMP.ADRS/@1,OS/WITH.OS"
:1657
:1658 RETURN      "SCTL/RETURN"
:1659 RETURN+1     "SCTL/RETURN+1"
:1660 RETURN+1.IF(MEM.REF.OK) "SCTL/RET.NOERR"
:1661
:1662 LOOP.IF(NEQ)  "SCTL/JMP.Z.CLR"
:1663 LOOP.IF(GEQU) "SCTL/JMP.C.SET"
:1664 LOOP.IF(NO INTERRUPT AND NO SYNC) ELSE SKIP.IF(SYNC) "SCTL/LOOP.IF.NO.I.AND.SYNC"
:1665
:1666 SKIP.IF[]     "SCTL/@1"
:1667 SKIP          "SCTL/SKIP"
  
```

```

:1668 .PAGE
:1669 :
:1670 : INSTRUCTION STREAM MACROS
:1671 :
:1672 : * THIS MACRO IS NOT FOR GENERAL USE!!!
:1673 : * THIS IS ONLY FOR USE IN THE FOLLOWING MACROS
:1674 :
:1675 :
:1676 : DECODE.SPEC ADRS[] 'DEC[21],IFUNC/SPEC,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/S
:1677 : DECODE.ASRC ADRS[] 'DEC[21],IFUNC/ASRC,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/A
:1678 : DECODE.ESRC ADRS[] 'DEC[21],IFUNC/ESRC,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/E
:1679 : DECODE.VSRC ADRS[] 'DEC[21],IFUNC/VSRC,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/V
:1680 : DECODE.FLOAT ADRS[] 'DEC[21],IFUNC/FLOAT,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/
:1681 : DECODE.OPC1 ADRS[] 'DEC[21],IFUNC/VAX.IRD,R.DST/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,OPC.SPEC/VAX.IRD
:1682 : EXEC.DISPATCH ADRS[] 'DEC[21],IFUNC/VAX.EXEC,IB.REQ/NOP,R.DST/NOP,CM.HI/LOW.BYTE,JCTL/JSR,OPC.SPEC/VAX.EXEC
:1683 :
:1684 : DECODE.CM.OPC ADRS[] 'DEC[21],IFUNC/CM.IRD,CM.HI/HI.BYTE,OPC.SPEC/CM.IRD,LD.OS/LOAD.OS
:1685 : DECODE.CM.DST ADRS[] 'DEC[21],IFUNC/CM.DST,OPC.SPEC/CM.DST,LD.OS/LOAD.OS,R.DST/ENAB
:1686 : DECODE.CM.SINGLE ADRS[] 'DEC[21],IFUNC/CM.SINGLE,OPC.SPEC/CM.SINGLE
:1687 : CM.EXEC ADRS[] 'DEC[21],IFUNC/CM.EXEC,JCTL/JUMP,OPC.SPEC/CM.EXEC,LD.OS/LOAD.OS
:1688 :
:1689 : SAVE.PC WRO 'BPC/SAVE.PC
:1690 : SAVE.PC WR4 'BPC/SAVE.PC
:1691 : SAVE.CM.PC WRO 'BPC/SAVE.PC
:1692 : SAVE.CM.PC WR4 'BPC/SAVE.PC
:1693 :
:1694 :
:1695 : THESE SKIP CONDITIONS ARE USED FOR THE DECODE MICRO INSTRUCTION
:1696 :
:1697 : SKIP.IF(1B VALID) 'JCTL/NO.JUMP.TST
:1698 : JMP.IF(1B VALID) 'JCTL/JUMP.VALID
:1699 : JSR.IF(1B VALID) 'JCTL/JSR.VALID
:1700 :
:1701 : THESE SKIP CONDITIONS ARE TO BE USED WITH THE COMPATIBILITY MODE DECODE
:1702 : MACROS.
:1703 :
:1704 : JMP.TO [] 'JCTL/JUMP,DEC[21]
:1705 : JSR.TO [] 'JCTL/JSR,DEC[21]
:1706 .BIN
  
```


ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MACRO DEFINITIONS 3-MAR-82
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Rev 01.00
MACRO DEFINITIONS VER 00.02

Fiche 1 Frame 14

Sequence 47

Page 41

```
:1707 .PAGE
:1708 .TOC 'FIELD DEFINITIONS FOR DATA PATH CONTROL MICRO WORD VER 00.01'
:1709
:1710 ; THIS SECTION IS FOR THE DATA PATH CONTROL ROMS
:1711 ;
:1712
:1713 .CCODE
:1714 SDP/=<8:0>,.ADDRESS,.VALIDITY=0
:1715
:1716 ; THIS FIELD CORRESPONDS TO THE 2901 ALU FUNCTION CONTROL PINS ... 15,14,13
:1717
:1718 ALU/=<2:0>,.DEFAULT=<ALU/R.OR.S>
:1719
:1720 R.PLUS.S=0 ; ADD R WITH S
:1721 S.MINUS.R=1 ; SUBTRACT R FROM S
:1722 R.MINUS.S=2 ; SUBTRACT S FROM R
:1723 R.OR.S=3 ; OR R WITH S
:1724 R.AND.S=4 ; AND R WITH S
:1725 NOTR.AND.S=5 ; COMPLEMENT R THEN AND WITH S
:1726 R.XOR.S=6 ; XOR R WITH S
:1727 R.XNOR.S=7 ; OR R WITH S THEN COMPLEMENT THE RESULT
:1728 SRC/=<8:6>,.DEFAULT=<SRC/D.0>
:1729
:1730 O.Q=2 ; RMX=0 SMX=Q
:1731 O.B=3 ; RMX=0 SMX=B
:1732 A.Q=0 ; RMX=A SMX=Q
:1733 A.B=01 ; RMX=A SMX=B
:1734 D.Q=06 ; RMX=D SMX=Q
:1735 D.O=7 ; RMX=D SMX=0
:1736 O.A=4 ; RMX=0 SMX=A
:1737 D.A=5 ; RMX=D SMX=A
:1738
:1739 ; THIS FIELD CORRESPONDES TO THE 2901 ALU DESTINATION
:1740 ; CONTROL PINS 18, 17 AND 16.
:1741
:1742 DST/=<5:3>,.DEFAULT=<DST/NOP>
:1743
:1744 LOAD.Q=0 ; LOAD Q-REGISTER
:1745 NOP=1 ; HOLD ALL REGISTERS
:1746 WRITE.B.A=2 ; WRITE RAM B-PORT AND OUTPUT RAM A-PORT
:1747 WRITE.B.F=3 ; WRITE RAM B-PORT AND OUTPUT ALU
:1748 RSHF.RAM.Q=4 ; RIGHT SHIFT RAM AND Q
:1749 RSHF.RAM=5 ; RIGHT SHIFT RAM
:1750 LSHF.RAM.Q=6 ; LEFT SHIFT RAM AND Q
:1751 LSHF.RAM=7 ; LEFT SHIFT RAM
:1752
:1753 CIN/=<9>,.DEFAULT=0
:1754
:1755 NO.CIN=0 ; NO CARRY IN TO ALU
:1756 CIN=1 ; CARRY IN TO ALU
:1757
:1758
:1759
```

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

FIELD DEFINITIONS J 4
MICRO2 1L(03) 3-MAR-82 10:02:46
FIELD DEFINITIONS FOR DATA PATH CONTROL MICRO WORD

Fiche 1 Frame J4
ENKCB Micro-diagnostic for DAP module
VER 00.01

Sequence 48
Rev 01.00 Page 42

```
:1760 .PAGE
:1761 .TOC "CONTROL ROM CONTENTS
:1762 .SEQUENTIAL
:1763 .THESE ARE THE DATA PATH CONTROL ROM CONTENTS
:1764
:1765 .THE FOLLOWING LOCATIONS ARE USED BY THE DECODE.IS MICRO INSTRUCTION
:1766
:1767 .THIS INSTRUCTION WILL DO PC+1 OPERATIONS
:1768
:1769 .BACK UP PC IN 2901 RAM
:1770 000:
:1771 DECODE.IS:
C 0000, 3D8 :1772 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0001, 3D8 :1773 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0002, 3D8 :1774 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0003, 3D8 :1775 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0004, 3D8 :1776 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0005, 3D8 :1777 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0006, 3D8 :1778 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0007, 3D8 :1779 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0008, 3D8 :1780 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0009, 3D8 :1781 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 000A, 3D8 :1782 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 000B, 3D8 :1783 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 000C, 3D8 :1784 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 000D, 3D8 :1785 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 000E, 3D8 :1786 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 000F, 3D8 :1787 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
:1788
:1789 .
:1790 .
:1791 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0010, 3C8 :1792 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0011, 3C8 :1793 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0012, 3C8 :1794 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0013, 3C8 :1795 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0014, 3C8 :1796 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0015, 3C8 :1797 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0016, 3C8 :1798 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0017, 3C8 :1799 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0018, 3C8 :1800 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0019, 3C8 :1801 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 001A, 3C8 :1802 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 001B, 3C8 :1803 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 001C, 3C8 :1804 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 001D, 3C8 :1805 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 001E, 3C8 :1806 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 001F, 3C8
```

```

:1807 .PAGE
:1808 : THESE ADDRESSES ARE USED BY THE 'JUMP' MICRO INSTRUCTION
:1809 :
:1810 : THE FOLLOWING CONTROL CODES WILL INSURE THAT NO DATA IS DESTROYED WITHIN
:1811 : THE 2901 DURING JUMP OR JSR OPERATIONS.
:1812
:1813 020:
:1814 JUMP:
C 0020, 3CB :1815 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0021, 3CB :1816 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0022, 3CB :1817 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0023, 3CB :1818 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0024, 3CB :1819 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0025, 3CB :1820 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0026, 3CB :1821 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0027, 3CB :1822 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0028, 3CB :1823 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0029, 3CB :1824 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 002A, 3CB :1825 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 002B, 3CB :1826 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 002C, 3CB :1827 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 002D, 3CB :1828 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 002E, 3CB :1829 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 002F, 3CB :1830 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0030, 3CB :1831 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0031, 3CB :1832 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0032, 3CB :1833 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0033, 3CB :1834 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0034, 3CB :1835 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0035, 3CB :1836 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0036, 3CB :1837 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0037, 3CB :1838 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0038, 3CB :1839 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0039, 3CB :1840 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 003A, 3CB :1841 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 003B, 3CB :1842 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 003C, 3CB :1843 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 003D, 3CB :1844 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 003E, 3CB :1845 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 003F, 3CB :1846 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN

```

```

:1847 .PAGE
:1848 : THESE LOCATIONS ARE USED BY THE MISC' MICRO INSTRUCTION
:1849 :
:1850 : THE FOLLOWING CONTROL CODES WILL INSURE THAT NO DATA IS DESTROYED WITHIN
:1851 : THE 2901 DURING MISC INSTRUCTION EXECUTION.
:1852 :
:1853 040:
:1854 MISC:
C 0040. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0041. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0042. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0043. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0044. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0045. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0046. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0047. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0048. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0049. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 004A. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 004B. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 004C. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 004D. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 004E. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 004F. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0050. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0051. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0052. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0053. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0054. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0055. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0056. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0057. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0058. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0059. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 005A. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 005B. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 005C. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 005D. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 005E. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 005F. 313 SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN

```

```

:1887 .PAGE
:1888 : THESE ADDRESSES ARE USED BY THE MEM.REQ MICRO INSTRUCTION
:1889 :
:1890 : THIS INSTRUCTION CAUSES WR[5] TO BE LOADED WITH THE VIRTUAL ADDRESS
:1891 : OF THE MEMORY REFERENCE.
:1892 060:
:1893 MEM.REQ:
C 0060, 3DB :1894 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0061, 3DB :1895 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0062, 3DB :1896 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0063, 3DB :1897 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0064, 3DB :1898 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0065, 3DB :1899 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0066, 3DB :1900 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0067, 3DB :1901 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0068, 3DB :1902 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0069, 3DB :1903 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 006A, 3DB :1904 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 006B, 3DB :1905 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 006C, 3DB :1906 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 006D, 3DB :1907 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 006E, 3DB :1908 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 006F, 3DB :1909 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0070, 3DB :1910 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0071, 3DB :1911 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0072, 3DB :1912 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0073, 3DB :1913 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0074, 3DB :1914 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0075, 3DB :1915 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0076, 3DB :1916 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0077, 3DB :1917 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0078, 3DB :1918 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0079, 3DB :1919 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 007A, 3DB :1920 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 007B, 3DB :1921 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 007C, 3DB :1922 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 007D, 3DB :1923 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 007E, 3DB :1924 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 007F, 3DB :1925 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN

```

	:1926	.PAGE
	:1927	: THESE ARE THE ADDRESSES FOR THE EXTENDED MICRO INSTRUCTION
	:1928	: ADDRESSES START AT 80-FF
	:1929	080:
	:1930	
C 0080, 118	:1931	WR.PLUS.O.TO.WR:
	:1932	SRC/O.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0081, 318	:1933	WR.INC.WR:
	:1934	SRC/O.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0082, 31A	:1935	WR.NEG.WR:
	:1936	SRC/O.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
C 0083, 31F	:1937	WR.COM.WR:
	:1938	SRC/O.A,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN
	:1939	
C 0084, 119	:1940	WR.DEC.WR:
	:1941	SRC/O.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
C 0085, 10B	:1942	FILL85:
	:1943	SRC/O.A
C 0086, 29B	:1944	MOV.Q.WR:
	:1945	SRC/O.Q,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0087, 08B	:1946	FILL87:
	:1947	SRC/O.Q
	:1948	
C 0088, 060	:1949	COND.ADD&SHIFT:
	:1950	SRC/A.B,ALU/R.PLUS.S,DST/RSHF.RAM.Q,CIN/NO.CIN
C 0089, 261	:1951	COND.SUB&SHIFT:
	:1952	SRC/A.B,ALU/S.MINUS.R,DST/RSHF.RAM.Q,CIN/CIN
C 008A, 04B	:1953	FILL8A:
	:1954	SRC/A.B
C 008B, 078	:1955	SHL2:
	:1956	SRC/A.B,ALU/R.PLUS.S,DST/LSHF.RAM,CIN/NO.CIN
	:1957	
C 008C, 0E3	:1958	ASHRC:
	:1959	SRC/O.B,ALU/R.OR.S,DST/RSHF.RAM.Q,CIN/NO.CIN
C 008D, 2EB	:1960	ASHR:
	:1961	SRC/O.B,ALU/R.OR.S,DST/RSHF.RAM,CIN/CIN
C 008E, 2F3	:1962	ASHLC:
	:1963	SRC/O.B,ALU/R.OR.S,DST/LSHF.RAM.Q,CIN/CIN
C 008F, 2FB	:1964	ASHL:
	:1965	SRC/O.B,ALU/R.OR.S,DST/LSHF.RAM,CIN/CIN
	:1966	
C 0090, 088	:1967	Q.PLUS.O:
	:1968	SRC/O.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 0091, 08B	:1969	FILL91:
	:1970	SRC/O.Q
C 0092, 20A	:1971	WR.CMP.Q:
	:1972	SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 0093, 209	:1973	Q.CMP.WR:
	:1974	SRC/A.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
	:1975	
C 0094, 081	:1976	DEC.Q:
	:1977	SRC/O.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/NO.CIN
C 0095, 280	:1978	INC.Q:
	:1979	SRC/O.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/CIN
	:1980	NEG.Q:

ZZ-ENKCB-1.0
ENKCB.MIC
ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) CONTROL ROM CONTENT 3-MAR-82
CONTROL ROM CONTENTS

Fiche 1 Frame B5

Sequence 53

ENKCB Micro-diagnostic for DAP module
VER 00.01

Rev 01.00

Page 47

C 0096, 282 :1981 SRC/0.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 0097, 287 :1982 COM.Q:
:1983 SRC/0.Q,ALU/R.XNOR.S,DST/LOAD.Q,CIN/CIN
:1984
C 0098, 04C :1985 WR.BIT.WR:
:1986 SRC/A.B,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
:1987 WR.CMP.WR:
C 0099, 24A :1988 SRC/A.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
:1989 FILL9A:
C 009A, 04B :1990 SRC/A.B
:1991 WR.PLUS.WR+1:
C 009B, 258 :1992 SRC/A.B,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
:1993
:1994 FILL9C:
C 009C, 04B :1995 SRC/A.B
:1996 FILL9D:
C 009D, 04B :1997 SRC/A.B
:1998 FILL9E:
C 009E, 0CB :1999 SRC/0.B
:2000 FILL9F:
C 009F, 0CB :2001 SRC/0.B
:2002
:2003 WR.PLUS.Q:
C 00A0, 000 :2004 SRC/A.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
:2005 WR.FROM.Q:
C 00A1, 201 :2006 SRC/A.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
:2007 Q.FROM.WR.Q:
C 00A2, 202 :2008 SRC/A.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
:2009 WR.OR.Q:
C 00A3, 203 :2010 SRC/A.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
:2011
:2012 WR.AND.Q:
C 00A4, 004 :2013 SRC/A.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
:2014 WR.MASK.Q:
C 00A5, 205 :2015 SRC/A.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/CIN
:2016 WR.XOR.Q:
C 00A6, 206 :2017 SRC/A.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
:2018 FILLA7:
C 00A7, 00B :2019 SRC/A.Q
:2020
:2021 WR.PLUS.WR.Q:
C 00A8, 040 :2022 SRC/A.B,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
:2023 WR.FROM.WR.Q:
C 00A9, 241 :2024 SRC/A.B,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
:2025 FILLAA:
C 00AA, 04B :2026 SRC/A.B
:2027 WR.OR.WR.Q:
C 00AB, 243 :2028 SRC/A.B,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
:2029
:2030 WR.AND.WR.Q:
C 00AC, 044 :2031 SRC/A.B,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
:2032 WR.MASK.WR.Q:
C 00AD, 245 :2033 SRC/A.B,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/CIN
:2034 WR.XOR.WR.Q:
C 00AE, 246 :2035 SRC/A.B,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) CONTROL ROM CONTENT 3-MAR-82
CONTROL ROM CONTENTS

Fiche 1 Frame C5

Sequence 54

ENKCB Micro-diagnostic for DAP module
VER 00.01

Rev 01.00

Page 48

C 00AF, 04B :2036 FILLAF:
:2037 SRC/A.B
:2038
C 00B0, 018 :2039 Q.PLUS.WR:
:2040 SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
:2041 WR.FROM.Q.WR:
C 00B1, 219 :2042 SRC/A.Q,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
:2043 Q.FROM.WR:
C 00B2, 21A :2044 SRC/A.Q,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
:2045 Q.OR.WR:
C 00B3, 21B :2046 SRC/A.Q,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
:2047
:2048 Q.AND.WR:
C 00B4, 01C :2049 SRC/A.Q,ALU/R.AND.S,DST/WRITE.B.F,CIN/NO.CIN
:2050 FILLB5:
C 00B5, 00B :2051 SRC/A.Q
:2052 Q.XOR.WR:
C 00B6, 21E :2053 SRC/A.Q,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
:2054 Q.BIT.WR:
C 00B7, 20C :2055 SRC/A.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
:2056
:2057 WR.PLUS.WR:
C 00B8, 058 :2058 SRC/A.B,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
:2059 WR.FROM.WR:
C 00B9, 259 :2060 SRC/A.B,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
:2061 WR.FROM.WR.WR:
C 00BA, 25A :2062 SRC/A.B,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
:2063 WR.OR.WR:
C 00BB, 25B :2064 SRC/A.B,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
:2065
:2066 WR.FROM.WR-1:
C 00BC, 059 :2067 SRC/A.B,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
:2068 WR.MASK.WR:
C 00BD, 25D :2069 SRC/A.B,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
:2070 WR.XOR.WR:
C 00BE, 25E :2071 SRC/A.B,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
:2072 WR.AND.WR:
C 00BF, 25C :2073 SRC/A.B,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
:2074

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) CONTROL ROM CONTENTS
3-MAR-82 10:02:46
D 5
3-MAR-82

Fiche 1 Frame D5

ENKCB Micro-diagnostic for DAP module
VER 00.01

Sequence 55
Rev 01.00
Page 49

```
:2075 .PAGE
:2076 : THIS IS THE DP CODES FOR THE 'MOVE' MICRO INSTRUCTION
:2077
:2078 0C0:
:2079
:2080 MOV.MEM.WR:
:2081 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
:2082 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
:2083 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
:2084 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
:2085 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
:2086 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
:2087 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
:2088 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
:2089
:2090 MOV.LS.MEM:
:2091 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
:2092 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
:2093 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
:2094 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
:2095 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
:2096 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
:2097 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
:2098 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
:2099
:2100 MOV.XWR.Q:
:2101 SRC/O.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
:2102 SRC/O.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
:2103 SRC/O.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
:2104 SRC/O.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
:2105 SRC/O.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
:2106 SRC/O.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
:2107 SRC/O.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
:2108 SRC/O.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
:2109
:2110 MOV.LS.WR:
:2111 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
:2112 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
:2113 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
:2114 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
:2115 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
:2116 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
:2117 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
:2118 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
:2119
```

C 00C0, 1D3
C 00C1, 1D3
C 00C2, 1D3
C 00C3, 1D3
C 00C4, 1D3
C 00C5, 1D3
C 00C6, 1D3
C 00C7, 1D3

C 00C8, 1CB
C 00C9, 1CB
C 00CA, 1CB
C 00CB, 1CB
C 00CC, 1CB
C 00CD, 1CB
C 00CE, 1CB
C 00CF, 1CB

C 00D0, 0C3
C 00D1, 0C3
C 00D2, 0C3
C 00D3, 0C3
C 00D4, 0C3
C 00D5, 0C3
C 00D6, 0C3
C 00D7, 0C3

C 00D8, 1DB
C 00D9, 1DB
C 00DA, 1DB
C 00DB, 1DB
C 00DC, 1DB
C 00DD, 1DB
C 00DE, 1DB
C 00DF, 1DB

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) CONTROL ROM CONTENT 3-MAR-82
CONTROL ROM CONTENTS

E 5

Fiche 1 Frame E5

Sequence 56

3-MAR-82 10:02:46 ENKCB Micro-diagnostic for CAP module Rev 01.00 Page 50
VER 00.01

```
:2120 .PAGE
:2121 MOV.MEM.LS:
C 00E0, 1CB :2122 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E1, 1CB :2123 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E2, 1CB :2124 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E3, 1CB :2125 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E4, 1CB :2126 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E5, 1CB :2127 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E6, 1CB :2128 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E7, 1CB :2129 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
:2130 MOV.ACC.LS:
C 00E8, 1CB :2131 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E9, 1CB :2132 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00EA, 1CB :2133 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00EB, 1CB :2134 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00EC, 1CB :2135 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00ED, 1CB :2136 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00EE, 1CB :2137 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00EF, 1CB :2138 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
:2139
:2140
:2141 WR.PLUS.OS:
C 00F0, 148 :2142 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F1, 148 :2143 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F2, 148 :2144 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F3, 148 :2145 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F4, 148 :2146 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F5, 148 :2147 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F6, 148 :2148 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F7, 148 :2149 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
:2150
:2151 MOV.WR.LS:
C 00F8, 10B :2152 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00F9, 10B :2153 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00FA, 10B :2154 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00FB, 10B :2155 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00FC, 10B :2156 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00FD, 10B :2157 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00FE, 10B :2158 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00FF, 10B :2159 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
:2160
```

```

:2161 .PAGE
:2162 : THIS GROUP OF INSTRUCTIONS IS FOR THE 'BASIC' MICRO INSTRUCTION
:2163 : THIS USES ADDRESSES 100-1FF
:2164
:2165 100:
:2166 LS.PLUS.WR:
C 0100, 158 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0101, 158 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0102, 158 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0103, 158 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
:2170
:2171 LS.FROM.WR:
C 0104, 359 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 0105, 359 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 0106, 359 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 0107, 359 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
:2176
:2177 LS.AND.WR:
C 0108, 35C SRC/D.A,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
C 0109, 35C SRC/D.A,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
C 010A, 35C SRC/D.A,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
C 010B, 35C SRC/D.A,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
:2180
:2181 LS.XOR.WR:
C 010C, 35E SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
C 010D, 35E SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
C 010E, 35E SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
C 010F, 35E SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
:2186
:2187 LS.FROM.WR-1:
C 0110, 159 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
C 0111, 159 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
C 0112, 159 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
C 0113, 159 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
:2192
:2193 LS.MASK.WR:
C 0114, 35D SRC/D.A,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
C 0115, 35D SRC/D.A,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
C 0116, 35D SRC/D.A,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
C 0117, 35D SRC/D.A,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
:2197
:2198 LS.PLUS.WR+1:
C 0118, 358 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0119, 358 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 011A, 358 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 011B, 358 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
:2202
:2203 LS.OR.WR:
C 011C, 35B SRC/D.A,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 011D, 35B SRC/D.A,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 011E, 35B SRC/D.A,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 011F, 35B SRC/D.A,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
:2207
:2208 WR.PLUS.LS.Q:
C 0120, 140 SRC/D.A,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 0121, 140 SRC/D.A,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 0122, 140 SRC/D.A,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 0123, 140 SRC/D.A,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
:2213
:2214 LS.FROM.WR.Q:
C 0124, 341 SRC/D.A,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 0125, 341 SRC/D.A,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN

```

C 0126, 341	:2216	SRC/D.A,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 0127, 341	:2217	SRC/D.A,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
	:2218	WR.FROM.LS.Q:
C 0128, 342	:2219	SRC/D.A,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 0129, 342	:2220	SRC/D.A,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 012A, 342	:2221	SRC/D.A,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 012B, 342	:2222	SRC/D.A,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
	:2223	WR.OR.LS.Q:
C 012C, 343	:2224	SRC/D.A,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 012D, 343	:2225	SRC/D.A,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 012E, 343	:2226	SRC/D.A,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 012F, 343	:2227	SRC/D.A,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
	:2228	
	:2229	WR.AND.LS.Q:
C 0130, 144	:2230	SRC/D.A,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 0131, 144	:2231	SRC/D.A,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 0132, 144	:2232	SRC/D.A,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 0133, 144	:2233	SRC/D.A,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
	:2234	WR.XOR.LS.Q:
C 0134, 346	:2235	SRC/D.A,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 0135, 346	:2236	SRC/D.A,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 0136, 346	:2237	SRC/D.A,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 0137, 346	:2238	SRC/D.A,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
	:2239	LS.CMP.WR:
C 0138, 34A	:2240	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 0139, 34A	:2241	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 013A, 34A	:2242	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 013B, 34A	:2243	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
	:2244	WR.CMP.LS:
C 013C, 349	:2245	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 013D, 349	:2246	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 013E, 349	:2247	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 013F, 349	:2248	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
	:2249	
	:2250	LS.PLUS.Q:
C 0140, 180	:2251	SRC/D.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 0141, 180	:2252	SRC/D.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 0142, 180	:2253	SRC/D.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 0143, 180	:2254	SRC/D.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
	:2255	LS.FROM.Q:
C 0144, 381	:2256	SRC/D.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 0145, 381	:2257	SRC/D.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 0146, 381	:2258	SRC/D.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 0147, 381	:2259	SRC/D.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
	:2260	Q.FROM.LS.Q:
C 0148, 382	:2261	SRC/D.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 0149, 382	:2262	SRC/D.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 014A, 382	:2263	SRC/D.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 014B, 382	:2264	SRC/D.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
	:2265	LS.OR.Q:
C 014C, 383	:2266	SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 014D, 383	:2267	SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 014E, 383	:2268	SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 014F, 383	:2269	SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
	:2270	

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) CONTROL ROM CONTENTS
3-MAR-82 10:02:46

H 5

CONTROL ROM CONTENT 3-MAR-82

Fiche 1 Frame H5

Sequence 59

ENKCB Micro-diagnostic for DAP module
VER 00.01

Rev 01.00

Page 53

```
:2271 LS.MASK.Q:
C 0150, 185 :2272 SRC/D.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 0151, 185 :2273 SRC/D.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 0152, 185 :2274 SRC/D.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 0153, 185 :2275 SRC/D.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/NO.CIN
:2276 LS.XOR.Q:
C 0154, 386 :2277 SRC/D.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 0155, 386 :2278 SRC/D.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 0156, 386 :2279 SRC/D.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 0157, 386 :2280 SRC/D.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
:2281 LS.AND.Q:
C 0158, 384 :2282 SRC/D.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/CIN
C 0159, 384 :2283 SRC/D.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/CIN
C 015A, 384 :2284 SRC/D.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/CIN
C 015B, 384 :2285 SRC/D.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/CIN
:2286 LS.BIT.Q:
C 015C, 38C :2287 SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
C 015D, 38C :2288 SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
C 015E, 38C :2289 SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
C 015F, 38C :2290 SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
:2291 MOV.LS.Q:
C 0160, 1C3 :2292 SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 0161, 1C3 :2293 SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 0162, 1C3 :2294 SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 0163, 1C3 :2295 SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
:2296 WR.BIT.LS:
C 0164, 34C :2297 SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 0165, 34C :2298 SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 0166, 34C :2299 SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 0167, 34C :2300 SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
:2301 LS.CMP.Q:
C 0168, 38A :2302 SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 0169, 38A :2303 SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 016A, 38A :2304 SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 016B, 38A :2305 SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
:2306 Q.CMP.LS:
C 016C, 389 :2307 SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 016D, 389 :2308 SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 016E, 389 :2309 SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 016F, 389 :2310 SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
:2311 Q.PLUS.LS.TO.WR:
:2312 SRC/D.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0170, 198 :2313 SRC/D.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0171, 198 :2314 SRC/D.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0172, 198 :2315 SRC/D.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0173, 198 :2316 SRC/D.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
:2317 WRA.FROM.LS.WRB:
C 0174, 35A :2318 SRC/D.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
C 0175, 35A :2319 SRC/D.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
C 0176, 35A :2320 SRC/D.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
C 0177, 35A :2321 SRC/D.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
:2322 LS.NEG.WR:
C 0178, 3D9 :2323 SRC/D.Q,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 0179, 3D9 :2324 SRC/D.Q,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
:2325
```

C 017A, 3D9	:2326	SRC/D.0,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 017B, 3D9	:2327	SRC/D.0,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
	:2328	LS.COM.WR:
C 017C, 3DF	:2329	SRC/D.0,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN
C 017D, 3DF	:2330	SRC/D.0,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN
C 017E, 3DF	:2331	SRC/D.0,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN
C 017F, 3DF	:2332	SRC/D.0,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) CONTROL ROM CONTENT 3-MAR-82
3-MAR-82 10:02:46
CONTROL ROM CONTENTS

Fiche 1 Frame J5

Sequence 61

ENKCB Micro-diagnostic for DAP module Rev 01.00 Page 55
VER 00.01

```
:2333 .PAGE
:2334
:2335 : THE FOLLOWING LOCATIONS HAVE THE LOCAL STORE AS THEIR DESTINATION.
:2336
:2337 : THIS FACT IS USED BY HARDWARE TO GENERATE LOCAL STORE WRITE PULSES
:2338
:2339 180:
:2340
:2341 LS.PLUS.WR.XCHG:
:2342 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
:2343 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
:2344 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
:2345 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
:2346 LS.FROM.WR.XCHG:
:2347 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.A,CIN/CIN
:2348 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.A,CIN/CIN
:2349 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.A,CIN/CIN
:2350 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.A,CIN/CIN
:2351 LS.AND.WR.XCHG:
:2352 SRC/D.A,ALU/R.AND.S,DST/WRITE.B.A,CIN/CIN
:2353 SRC/D.A,ALU/R.AND.S,DST/WRITE.B.A,CIN/CIN
:2354 SRC/D.A,ALU/R.AND.S,DST/WRITE.B.A,CIN/CIN
:2355 SRC/D.A,ALU/R.AND.S,DST/WRITE.B.A,CIN/CIN
:2356 LS.XOR.WR.XCHG:
:2357 SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.A,CIN/CIN
:2358 SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.A,CIN/CIN
:2359 SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.A,CIN/CIN
:2360 SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.A,CIN/CIN
:2361
:2362 SWAP.LS.WR:
:2363 SRC/D.O,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
:2364 SRC/D.O,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
:2365 SRC/D.O,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
:2366 SRC/D.O,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
:2367 CLR.LS:
:2368 SRC/D.O,ALU/R.AND.S,DST/NOP,CIN/CIN
:2369 SRC/D.O,ALU/R.AND.S,DST/NOP,CIN/CIN
:2370 SRC/D.O,ALU/R.AND.S,DST/NOP,CIN/CIN
:2371 SRC/D.O,ALU/R.AND.S,DST/NOP,CIN/CIN
:2372 MOV.Q.WR&WR.LS:
:2373 SRC/O.Q,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
:2374 SRC/O.Q,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
:2375 SRC/O.Q,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
:2376 SRC/O.Q,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
:2377 WR.PLUS.Q.XCHG:
:2378 SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
:2379 SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
:2380 SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
:2381 SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
:2382
:2383 WR.FROM.LS-1:
:2384 SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
:2385 SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
:2386 SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
:2387 SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
```

C 0180, 150
C 0181, 150
C 0182, 150
C 0183, 150

C 0184, 351
C 0185, 351
C 0186, 351
C 0187, 351

C 0188, 354
C 0189, 354
C 018A, 354
C 018B, 354

C 018C, 356
C 018D, 356
C 018E, 356
C 018F, 356

C 0190, 1D3
C 0191, 1D3
C 0192, 1D3
C 0193, 1D3

C 0194, 3CC
C 0195, 3CC
C 0196, 3CC
C 0197, 3CC

C 0198, 293
C 0199, 293
C 019A, 293
C 019B, 293

C 019C, 010
C 019D, 010
C 019E, 010
C 019F, 010

C 01A0, 14A
C 01A1, 14A
C 01A2, 14A
C 01A3, 14A

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) CONTROL ROM CONTENT 3-MAR-82
CONTROL ROM CONTENTS

K 5

Fiche 1 Frame K5

Sequence 62

ENKCB Micro-diagnostic for DAP module Rev 01.00 Page 56
VER 00.01

	:2388	LS.FROM.WR.LS:
C 01A4, 349	:2389	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01A5, 349	:2390	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01A6, 349	:2391	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01A7, 349	:2392	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
	:2393	WR.PLUS.LS+1:
C 01A8, 348	:2394	SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01A9, 348	:2395	SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01AA, 348	:2396	SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01AB, 348	:2397	SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/CIN
	:2398	WR.FROM.LS:
C 01AC, 34A	:2399	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01AD, 34A	:2400	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01AE, 34A	:2401	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01AF, 34A	:2402	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
	:2403	
	:2404	WR.PLUS.LS:
C 01B0, 148	:2405	SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01B1, 148	:2406	SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01B2, 148	:2407	SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01B3, 148	:2408	SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
	:2409	WR.OR.LS:
C 01B4, 34B	:2410	SRC/D.A,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01B5, 34B	:2411	SRC/D.A,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01B6, 34B	:2412	SRC/D.A,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01B7, 34B	:2413	SRC/D.A,ALU/R.OR.S,DST/NOP,CIN/CIN
	:2414	WR.AND.LS:
C 01B8, 34C	:2415	SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 01B9, 34C	:2416	SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 01BA, 34C	:2417	SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 01BB, 34C	:2418	SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
	:2419	WR.XOR.LS:
C 01BC, 34E	:2420	SRC/D.A,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 01BD, 34E	:2421	SRC/D.A,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 01BE, 34E	:2422	SRC/D.A,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 01BF, 34E	:2423	SRC/D.A,ALU/R.XOR.S,DST/NOP,CIN/CIN
	:2424	
	:2425	Q.PLUS.LS:
C 01C0, 188	:2426	SRC/D.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01C1, 188	:2427	SRC/D.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01C2, 188	:2428	SRC/D.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01C3, 188	:2429	SRC/D.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
	:2430	LS.FROM.Q.LS:
C 01C4, 389	:2431	SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01C5, 389	:2432	SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01C6, 389	:2433	SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01C7, 389	:2434	SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
	:2435	Q.FROM.LS:
C 01C8, 38A	:2436	SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01C9, 38A	:2437	SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01CA, 38A	:2438	SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01CB, 38A	:2439	SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
	:2440	Q.OR.LS:
C 01CC, 38B	:2441	SRC/D.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01CD, 38B	:2442	SRC/D.Q,ALU/R.OR.S,DST/NOP,CIN/CIN

22-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) CONTROL ROM CONTENT 3-MAR-82
CONTROL ROM CONTENTS

Fiche 1 Frame L5

Sequence 63

3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Rev 01.00 Page 57
VER 00.01

C 01CE, 38B	:2443	SRC/D.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01CF, 38B	:2444	SRC/D.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
	:2445	
	:2446	Q.AND.LS:
C 01D0, 18C	:2447	SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
C 01D1, 18C	:2448	SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
C 01D2, 18C	:2449	SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
C 01D3, 18C	:2450	SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
	:2451	Q.XOR.LS:
C 01D4, 38E	:2452	SRC/D.Q,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 01D5, 38E	:2453	SRC/D.Q,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 01D6, 38E	:2454	SRC/D.Q,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 01D7, 38E	:2455	SRC/D.Q,ALU/R.XOR.S,DST/NOP,CIN/CIN
	:2456	MOV.Q.LS:
C 01D8, 28B	:2457	SRC/O.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01D9, 28B	:2458	SRC/O.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01DA, 28B	:2459	SRC/O.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01DB, 28B	:2460	SRC/O.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
	:2461	Q.NEG.LS:
C 01DC, 28A	:2462	SRC/O.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01DD, 28A	:2463	SRC/O.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01DE, 28A	:2464	SRC/O.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01DF, 28A	:2465	SRC/O.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
	:2466	
	:2467	WR.COM.LS:
C 01E0, 0CF	:2468	SRC/O.B,ALU/R.XNOR.S,DST/NOP,CIN/NO.CIN
C 01E1, 0CF	:2469	SRC/O.B,ALU/R.XNOR.S,DST/NOP,CIN/NO.CIN
C 01E2, 0CF	:2470	SRC/O.B,ALU/R.XNOR.S,DST/NOP,CIN/NO.CIN
C 01E3, 0CF	:2471	SRC/O.B,ALU/R.XNOR.S,DST/NOP,CIN/NO.CIN
	:2472	WR.NEG.LS:
C 01E4, 2CA	:2473	SRC/O.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01E5, 2CA	:2474	SRC/O.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01E6, 2CA	:2475	SRC/O.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01E7, 2CA	:2476	SRC/O.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
	:2477	Q.PLUS.WR.TO.LS:
C 01E8, 008	:2478	SRC/A.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01E9, 008	:2479	SRC/A.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01EA, 008	:2480	SRC/A.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01EB, 008	:2481	SRC/A.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
	:2482	Q.FROM.WR.TO.LS:
C 01EC, 20A	:2483	SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01ED, 20A	:2484	SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01EE, 20A	:2485	SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01EF, 20A	:2486	SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
	:2487	
	:2488	DEC.LS:
C 01F0, 1CA	:2489	SRC/D.O,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 01F1, 1CA	:2490	SRC/D.O,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 01F2, 1CA	:2491	SRC/D.O,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 01F3, 1CA	:2492	SRC/D.O,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
	:2493	COM.LS:
C 01F4, 3CF	:2494	SRC/D.O,ALU/R.XNOR.S,DST/NOP,CIN/CIN
C 01F5, 3CF	:2495	SRC/D.O,ALU/R.XNOR.S,DST/NOP,CIN/CIN
C 01F6, 3CF	:2496	SRC/D.O,ALU/R.XNOR.S,DST/NOP,CIN/CIN
C 01F7, 3CF	:2497	SRC/D.O,ALU/R.XNOR.S,DST/NOP,CIN/CIN

	:2498	NEG.LS:	
C 01F8, 3C9	:2499		SRC/D.0,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01F9, 3C9	:2500		SRC/D.0,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01FA, 3C9	:2501		SRC/D.0,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01FB, 3C9	:2502		SRC/D.0,ALU/S.MINUS.R,DST/NOP,CIN/CIN
	:2503	INC.LS:	
C 01FC, 3C8	:2504		SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01FD, 3C8	:2505		SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01FE, 3C8	:2506		SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01FF, 3C8	:2507		SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
	:2508		
	:2509	.UCODE	

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

DIAGNOSTIC MACROS A 3-MAR-82
MICRO2 1L(03) 3-MAR-82 10:02:46
DIAGNOSTIC MACROS AND DEFINITIONS

Fiche 1 Frame N5

Sequence 65
Rev 01.00

Page 59

```
:2510 .PAGE 'DIAGNOSTIC MACROS AND DEFINITIONS'
:2511
:2512 D.ADRS/= <15:9> , .VALIDITY=<D.VAL> , .DEFAULT=0
:2513
:2514
:2515 SAV.WR0=1 ; STORAGE FOR WR0
:2516 SAV.WR1=2 ; STORAGE FOR WR1
:2517 SAV.WR2=3 ; STORAGE FOR WR2
:2518 SAV.WR3=4 ; STORAGE FOR WR3
:2519 T5.SUB=5 ; RESERVED FOR COMMON SUBROUTINES
:2520 T6.SUB=6 ; RESERVED FOR COMMON SUBROUTINES
:2521 T7.SUB=7 ; RESERVED FOR COMMON SUBROUTINES
:2522 TRANSFER.POINTER=7 ; POINTER FOR TRANSFER ROUTINE
:2523 MEMORY.SIZE=7 ; STORAGE FOR MEMORY SIZE IN 256KB BLOCKS AFTER
:2524 ; SIZING
:2525 FPA.FOUND=7 ; STORAGE FOR RESULT OF SIZING FOR FPA PRESENT
:2526 UBE.FOUND=7 ; STORAGE FOR RESULT OF SIZING FOR UBE PRESENT
:2527 T8=8 ; TEMP LOCATION 8
:2528 T9=9 ; TEMP LOCATION 9
:2529 T10=0A ; TEMP LOCATION 10
:2530 T11=0B ; TEMP LOCATION 11
:2531 T12=0C ; TEMP LOCATION 12
:2532 T13=0D ; TEMP LOCATION 13
:2533 T14=0E ; TEMP LOCATION 14
:2534 T15=0F ; TEMP LOCATION 15
:2535 PC=10 ; MACRO PROGRAM COUNTER
:2536
:2537 WR.BACKUP=11 ; BACKUP WR FOR MOV MEM.DATA TO WRC]
:2538 MM.LASTADDR+1=12 ; STORAGE OF LAST ADDRESS IN MAIN MEMORY PLUS
:2539 ; 1 (FIRST NON-EXISTANT MEMORY ADDRESS)
:2540 #FF=13 ; MASK
:2541 #FFFF=14 ; MASK
:2542 #FF000000=15 ; MASK
:2543 #FFFFFF00=16 ; MASK
:2544 CSR0.MASK=16 ; MASK
:2545 CSR1.MASK=17 ; MASK (01003FFF)
:2546 CSR2.MASK=18 ; MASK (7FFE3FFF)
:2547 #FE7FFFFFFF=19 ; MASK
:2548 UBS.SET=1A ; CONSTANT (7FF80000) USED BY UBS ADDRESS TEST
:2549 UBS.DATA=1B ; MASK (7FFF8000) USED AS ERROR MASK FOR UBS
:2550 ; DATA TEST
:2551
:2552 ERR.CON.NA=1E ; CONSTANT OF 800500(H) BITS 23,10,8 SET FOR
:2553 ; LOADING CONTROL AND STATUS REG IN INITIAL
:2554 ; TESTS
:2555 EKR.CON=1F ; CONSTANT OF 500(H) BITS 10 & 8 SET FOR
:2556 ; LOADING CONTROL AND STATUS REG IN INITIAL
:2557 ; TESTS
:2558
:2559 #1=20 ; PATTERN 00000001 (HEX)
:2560 #2=21 ; PATTERN 00000002
:2561 #4=22 ; PATTERN 00000004
:2562 #8=23 ; PATTERN 00000008
:2563 #10=24 ; PATTERN 00000010
:2564 #20=25 ; PATTERN 00000020
```

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

DIAGNOSTIC MACROS A 3-MAR-82
MICRO2 1L(03) 3-MAR-82 10:02:46
DIAGNOSTIC MACROS AND DEFINITIONS

B 6

Fiche 1 Frame B6

Sequence 66

Rev 01.00

Page 60

ZZ
:
:

:2565	#40=26	: PATTERN 00000040
:2566	#80=27	: PATTERN 00000080
:2567	#100=28	: PATTERN 00000100
:2568	#200=29	: PATTERN 00000200
:2569	#400=2A	: PATTERN 00000400
:2570	#800=2B	: PATTERN 00000800
:2571	#1000=2C	: PATTERN 00001000
:2572	#2000=2D	: PATTERN 00002000
:2573	#4000=2E	: PATTERN 00004000
:2574	#8000=2F	: PATTERN 00008000
:2575	#10000=30	: PATTERN 00010000
:2576	#20000=31	: PATTERN 00020000
:2577	#40000=32	: PATTERN 00040000
:2578	#80000=33	: PATTERN 00080000
:2579	#100000=34	: PATTERN 00100000
:2580	#200000=35	: PATTERN 00200000
:2581	#400000=36	: PATTERN 00400000
:2582	#800000=37	: PATTERN 00800000
:2583	#1000000=38	: PATTERN 01000000
:2584	#2000000=39	: PATTERN 02000000
:2585	#4000000=3A	: PATTERN 04000000
:2586	#8000000=3B	: PATTERN 08000000
:2587	#10000000=3C	: PATTERN 10000000
:2588	#20000000=3D	: PATTERN 20000000
:2589	#40000000=3E	: PATTERN 40000000
:2590	#80000000=3F	: PATTERN 80000000
:2591		
:2592	BIT0=20	: PATTERN 00000001
:2593	BIT1=21	: PATTERN 00000002
:2594	BIT2=22	: PATTERN 00000004
:2595	BIT3=23	: PATTERN 00000008
:2596	BIT4=24	: PATTERN 00000010
:2597	BIT5=25	: PATTERN 00000020
:2598	BIT6=26	: PATTERN 00000040
:2599	BIT7=27	: PATTERN 00000080
:2600	BIT8=28	: PATTERN 00000100
:2601	BIT9=29	: PATTERN 00000200
:2602	BIT10=2A	: PATTERN 00000400
:2603	BIT11=2B	: PATTERN 00000800
:2604	BIT12=2C	: PATTERN 00001000
:2605	BIT13=2D	: PATTERN 00002000
:2606	BIT14=2E	: PATTERN 00004000
:2607	BIT15=2F	: PATTERN 00008000
:2608	BIT16=30	: PATTERN 00010000
:2609	BIT17=31	: PATTERN 00020000
:2610	BIT18=32	: PATTERN 00040000
:2611	BIT19=33	: PATTERN 00080000
:2612	BIT20=34	: PATTERN 00100000
:2613	BIT21=35	: PATTERN 00200000
:2614	BIT22=36	: PATTERN 00400000
:2615	BIT23=37	: PATTERN 00800000
:2616	BIT24=38	: PATTERN 01000000
:2617	BIT25=39	: PATTERN 02000000
:2618	BIT26=3A	: PATTERN 04000000
:2619	BIT27=3B	: PATTERN 08000000

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) 3-MAR-82 10:02:46
DIAGNOSTIC MACROS AND DEFINITIONS

Fiche 1 Frame C6

Sequence 67

Rev 01.00 Page 61

```
:2620      BIT28=3C      : PATTERN 10000000
:2621      BIT29=3D      : PATTERN 20000000
:2622      BIT30=3E      : PATTERN 40000000
:2623      BIT31=3F      : PATTERN 80000000
:2624
:2625
:2626      ;CSR ADDRESS MNEMONICS
:2627
:2628      CSR0=4E      : ALL BITS CLEAR TO ACCESS CSR 0
:2629      CSR1=22      : BIT 2 SET TO ACCESS CSR 1
:2630      CSR2=23      : BIT 3 SET TO ACCESS CSR 2
:2631
:2632      ;CONTROL AND STATUS WORD BITS
:2633
:2634      PRINT.UBE=26      : PRINT IF UBE PRESENT OR NOT PRESENT (BIT 6)
:2635                          : (INDICATOR IN LS 7)
:2636      PRINT.R80=27      : PRINT IF R80 PRESENT OR NOT PRESENT AND ON
:2637                          : WHICH DRIVE (BIT 7). INDICATOR IN LS 7,
:2638                          : DRIVE NUMBER IN LS 6.
:2639      ERROR=28          : TEST ERROR INDICATION (BIT 8)
:2640      SET.PA.ERR=29      : TELLS CONSOLE TO CREATE A PARITY ERROR AT WCS
:2641                          : ADDRESS POINTED TO BY LS 7 (BIT 9)
:2642      CONSOLE.LOE=2A      : INDICATES CONSOLE DOES ERROR LOOPING (BIT 10)
:2643                          : (FOR INITIAL TESTS)
:2644      PRINT.MEM.SIZE=2B    : PRINT MEMORY SIZE IN 256K BLOCKS (BIT 11)
:2645                          : (SIZE IN LS 7)
:2646      PRINT.FPA=2C      : PRINT IF FPA PRESENT OR NOT PRESENT (BIT 12)
:2647                          : (INDICATOR IN LS 7)
:2648      XFER.DATA=2D        : INDICATES A DATA TRANSFER TO LS OR MM (BIT 13)
:2649      EOS=2E            : END OF SECTION (BIT 14)
:2650      BEGIN.TEST=2F      : BEGINNING OF TEST (BIT 15)
:2651      INTERRUPT.EN=30     : ENABLE INTERRUPT OR SIGNAL SPECIFIED BY BITS
:2652                          : 17 THRU 22 AND 28 THRU 30 (BIT 16)
:2653
:2654      CONSOLE.HALT=31      : CONSOLE HALT INTERRUPT (BIT 17)
:2655      PWR.FAIL=32         : PWR FAIL INT (BIT 18)
:2656      INTERVAL.TIM=33    : INTRVL TIM INT (BIT 19)
:2657      CONSOLE.ATTN=34     : CONSOLE ATTN INTERRUPT (BIT 20)
:2658      CONSOLE.ACK=35     : CONSOLE ACK INTERRUPT (BIT 21)
:2659      DISABLE.MEM.REF=36  : DISABLE MEMORY REFERANCES (BIT 22)
:2660      CINIT=3C           : UNIBUS INIT SIGNAL TO THE MCT (BIT 28)
:2661      UBS.DCLO=3D        : UNIBUS DC LO INDICATION TO THE MCT (BIT 29)
:2662      UBS.BBSY=3E        : UNIBUS BUS BUSY INDICATOR TO MCT (BIT 30)
:2663
:2664      NA.EXP.REC=37        : PRINT N/A UNDER EXPECTED AND RECEIVED (BIT 23)
:2665      OTHER.DATA=38       : PRINT A VALUE UNDER OTHER DATA (BIT 24)
:2666      CONSOLE.LOOP=39     : CONSOLE PERFORMS A LOOP ACCORDING TO LOOP
:2667                          : COUNT IN LS (BIT 25)
:2668      PRINT.CRD=3A        : PRINT NUMBER OF SINGLE BIT ERRORS (BIT 26)
:2669      CLR.PA.ERR=3B       : TELLS CONSOLE TO CLEAR PARITY ERROR AT WCS
:2670                          : ADDRESS POINTED TO BY LS 7 (BIT 27)
:2671      PRINT.IDC=3F       : PRINT IF IDC PRESENT OR NOT PRESENT (BIT 31)
:2672                          : (INDICATOR IN LS 7)
:2673
:2674      ;MODULE CODES
```

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

DIAGNOSTIC MACROS A 3-MAR-82 D 6
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module
DIAGNOSTIC MACROS AND DEFINITIONS

Fiche 1 Frame D6

Sequence 68

Rev 01.00

Page 62

```
:2675
:2676      WCS=20      : MODULE CODE FOR WCS BOARD (BIT 0 SET)
:2677      CPU=21      : MODULE CODE FOR CPU BOARD (BIT 1 SET)
:2678      MCT=22      : MODULE CODE FOR MCT BOARD (BIT 2 SET)
:2679      FPA=23      : MODULE CODE FOR FPA BOARD (BIT 3 SET)
:2680      ARRAY=24    : MODULE CODE FOR ARRAY BOARD (BIT 4 SET)
:2681      IDC=25      : MODULE CODE FOR IDC BOARD (BIT 5 SET)
:2682
:2683 :CSR 1 ERROR BITS
:2684
:2685      VALID.ERR=2E  : VALID NOT SET IF 1 (BIT 14)
:2686      TB.PAR.ERR=2F : TB PARITY ERROR (BIT 15)
:2687      NXM=30        : NON-EXISTANT MEMORY ERROR (BIT 16)
:2688      UB.BSY=31     : UNIBUS BUSY (BIT 17)
:2689      ADP.REG=32    : UNIBUS ADAPTER REGISTER SELECT (BIT 18)
:2690      WR.ACROSS.PG=33 : WRITE ACROSS PAGE BOUNDARY ERROR (BIT 19)
:2691      OP.ERR=34     : OPERATION ERROR (BIT 20)
:2692      TB.MISS=35    : TB MISS (VIRTUAL ADDRESS TRANSLATION NOT IN
:2693                    : BUFFER) (BIT 21)
:2694      ACCESS.REFUSED=36 : PROTECTION ERROR ON ACCESS (BIT 22)
:2695      MODIFY.REFUSED=37 : PROTECTION ERROR ON WRITE (BIT 23)
:2696
:2697      CRD=3E         : SINGLE BIT ERROR CORRECTED (BIT 30)
:2698      RDS=3F        : UNCORRECTABLE DATA ERROR (BIT 31)
:2699
:2700
:2701 :CSR 1 CONTROL BITS
:2702
:2703      ECC.DIS=39     : CSR1 ECC DISABLE BIT (BIT 25)
:2704      DIAG.CHK=3A   : CSR1 DIAG CHK BIT (BIT 26)
:2705      MME=3B        : CSR1 MEMORY MANAGMENT ENABLE BIT (BIT 27)
:2706      INH.CRD=3C    : CSR1 INHIBIT REPORT CRD BIT (BIT 28)
:2707      TB.PAR.DIAG=3D : CSR1 TB PAR DIAG BIT (FORCE TB PARITY ERR)
:2708                    : (BIT 29)
:2709
:2710
:2711 :UBE CONTROL REGISTER BITS
:2712
:2713 :CONTROL REG 1
:2714      GO=20         : START UBE (BIT 0)
:2715      BR4=21        : ISSUE BUS REQUEST 4 (BIT 1)
:2716      BR5=22        : ISSUE BUS REQUEST 5 (BIT 2)
:2717      BR6=23        : ISSUE BUS REQUEST 6 (BIT 3)
:2718      BR7=24        : ISSUE BUS REQUEST 7 (BIT 4)
:2719      NPR=25        : ISSUE NPR (BIT 5)
:2720      INT.ODNE=26    : INTERRUPT ON DONE (WHEN CCOVF) (BIT 6)
:2721      RDY=27        : READY BIT, SET WHEN READY TO BEGIN FUNC (BIT 7)
:2722      C0=28         : C0 BIT (BIT 8)
:2723      C1=29         : C1 BIT (BIT 9)
:2724      FUNA=2A       : FUNCTION BIT A (BIT 10)
:2725      FUNB=2B       : FUNCTION BIT B (BIT 11)
:2726      CC+DAT=2C     : DATA FROM CYCLE COUNT IF SET, DATA REG IF CLEAR
:2727      INH.DATIP.ROL=2D : INHIBIT ROL ON DATIP (BIT 13)
:2728      DATOB.ON.DATIP=2E : DO DATOB AFTER DATIP CYCLE (BIT 14)
:2729      ERR=2F         : EXERCISOR OR UNIBUS ERROR (BIT 15)
```

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

DIAGNOSTIC MACROS A 3-MAR-82
MICRO2 1L(03) 3-MAR-82 10:02:46
DIAGNOSTIC MACROS AND DEFINITIONS

E 6

Fiche 1 Frame E6

Sequence 69

Rev 01.00

Page 63

```
:2730  
:2731 ;CONTROL REG 2  
:2732 EXT.MEM0=20 ; EXTENDED MEMORY BIT 0 (BIT 0)  
:2733 EXT.MEM1=21 ; EXTENDED MEMORY BIT 1 (BIT 1)  
:2734 INH.INC.ADDR&CC=22 ; INHIBIT INC ON CYCLE COUNT AND ADDR REG (BIT 2)  
:2735 INH.SACK=23 ; INHIBIT ASSERTING BUS SACK ON BUS GRANT (BIT 3)  
:2736 PWR.DWN.SEQ=24 ; ASSERT ACLO (BIT 4)  
:2737 WNG.GNT.BCK=25 ; NO BUS GRANT RECV TO HIGEST REQUEST (BIT 5)  
:2738 MAX.LATE.ERR=26 ; NPR AND BR LATENCY TIME EXCEEDED (BIT 6)  
:2739 NO.SACK.ERR=27 ; NO TIMEOUT AT CPU WHEN NO SACK (BIT 7)  
:2740 NO.SSYN.ERR=28 ; NO SSYN RECEIVED AFTER ASSERTING MSYN (BIT 8)  
:2741 WRG.A.LINES=29 ; ADDRESS ERR ON SENT & RECEIVED ADDRESS (BIT 9)  
:2742 NO.GNT+NOT.ONE.GNT=2A ; NO GRANT OR MORE THAN 1 GRANT AFTER REQ. (BIT 10)  
:2743 INTR.SSYN.ERR=2B ; NO SSYN RECEIVED AFTER BUS INTR (BIT 11)  
:2744 ENB.PB=2C ; ENABLE PARITY BIT B (BIT 12)  
:2745 CCOVF=2D ; CYCLE COUNT OVERFLOW (BIT 13)  
:2746 TM.DLY=2E ; REQUEST BUS EVERY 10 USEC (BIT 14)  
:2747  
:2748 ;BG6 MNEMONIC  
:2749 BG6=23 ; BIT 3 SET FOR BG 6 ADDRESS  
:2750  
:2751 ;ERROR AND CONTROL INFORMATION  
:2752  
:2753 CONTROL.STATUS=40 ; CONTROL AND STATUS WORD  
:2754 ERROR.NUMBER=41 ; ERROR NUMBER OF CURRENT TEST  
:2755 EXPECTED.DATA=42 ; EXPECTED RESULT OF CURRENT TEST  
:2756 RECEIVED.DATA=43 ; ACTUAL RESULT OF CURRENT TEST  
:2757 ADDRESS.DATA=44 ; OTHER PERTINENT DATA  
:2758 ERROR.MASK=45 ; BITS TO CHECK IN RESULT  
:2759 MODULE.NUM=46 ; STORAGE OF MODULE NUMBER (CODED)  
:2760 LOOP.CNT=47 ; STORAGE OF LOOP COUNT FOR CONSOLE LOOP  
:2761 CONTROL.  
:2762 ERROR.CONTROL=48 ; STORAGE FOR ERROR CONTROL (LOOP ON ERROR ETC.)  
:2763 LOE=20 ; LOOP ON ERROR IF SET (BIT0)  
:2764 NER=21 ; NO ERROR REPORT IF SET (BIT1)  
:2765 BELL=22 ; BELL ON ERROR IF SET (BIT2)  
:2766 HOE=23 ; HALT ON ERROR IF SET (BIT3)  
:2767 SER=24 ; ENABLE ERROR REPORT ON CRD ERRORS IF SET  
:2768 APT.PRESENT=25 ; APT PRESENT IF SET (BIT5)  
:2769 LOOP.COMMAND=26 ; LOOP COMMAND TYPED IF SET (BIT 6)  
:2770  
:2771 APT.PARAM=49 ; APT RUN TIME PARAMETER REGS (MEM SIZE, HARD-  
:2772 ; WARE CONFIG, SOFTWARE CONFIG IN BYTES 0, 1  
:2773 ; AND 2 RESPECTIVELY. BYTE 3 FOR FUTURE USE)  
:2774 APT.RESERVED=4A ; RESERVED FOR FUTURE APT USE  
:2775  
:2776 ;MISCELLANEOUS CONSTANTS AND STORAGE  
:2777  
:2778 %AAAAAAAA=4C ; CONSTANT  
:2779 %ALTER.ODD=4C ; CONSTANT  
:2780 #55555555=4D ; CONSTANT  
:2781 ALTER.EVEN=4D ; CONSTANT  
:2782 #0=4E ; CONSTANT  
:2783 ZERO=4E ; CONSTANT  
:2784 #FFFFFFFF=4F ; CONSTANT
```

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

DIAGNOSTIC MACROS A 3-MAR-82
MICRO2 1L(03) 3-MAR-82 10:02:46
DIAGNOSTIC MACROS AND DEFINITIONS

F 6

Fiche 1 Frame F6

Sequence 70

ENKCB Micro-diagnostic for DAP module

Rev 01.00

Page 64

```
:2785      ONES=4F      ; CONSTANT
:2786      PREVIOUS.ERROR=50 ; ERROR NUMBER OF LAST ERROR
:2787
:2788      DATA.1=51    ; STORAGE FOR FPA DATA
:2789      DATA.2=52    ; STORAGE FOR FPA DATA
:2790      DATA.3=53    ; STORAGE FOR FPA DATA
:2791      FIRST.FLT=54  ; STORAGE FOR FPA DATA
:2792      SEC.FLT=55    ; STORAGE FOR FPA DATA
:2793      THIRD.FLT=56  ; STORAGE FOR FPA DATA
:2794      T57=57        ; TEMP LOCATION 57
:2795      T58=58        ; TEMP LOCATION 58
:2796      T59=59        ; TEMP LOCATION 59
:2797
:2798      BEDB=5A        ;UBE (BUS EXERCISOR) DATA BUF REG
:2799      BECC=5B        ;UBE (BUS EXERCISOR) CYCLE COUNT REG
:2800      BEBA=5C        ;UBE (BUS EXERCISOR) ADDR REG
:2801      BECR1=5D       ;UBE (BUS EXERCISOR) CONTROL REG 1
:2802      CLEAR.ERR.ADDR=5E ;UBE CLEAR ERROR REG ADDRESS
:2803      BECR2=5F       ;UBE (BUS EXERCISOR) CONTROL REG 2
:2804
:2805      #3(H)=60       ; CONSTANT
:2806      #12(H)=61      ; CONSTANT
:2807      #33333333=62   ; CONSTANT
:2808      #0F0F0F0F=63   ; CONSTANT
:2809      #00FF00FF=64   ; CONSTANT
:2810      #162(H)=65     ; CONSTANT FOR LS TRANSFER POINTER
:2811      BR7.IDENT=66   ; BR7 IDENTIFIER (1010(B) IN BITS 5-2)
:2812      BG7=67         ; BG7 ADDRESS (BITS 2 AND 3 SET)
:2813
:2814      ;TEMPS FOR CPU TESTS
:2815      L30=30          ;LS 30 TEMP (RESTORED TO 10000 AFTER USE)
:2816      L31=31          ;LS 31 TEMP (RESTORE TO 20000 AFTER USE)
:2817      L53=53          ;LS 53 TEMP
:2818      L73=73          ;LS 73 TEMP
:2819
:2820      ;REGISTER ADDRESSES
:2821
:2822      CONTROL.OS=78     ; HARDWARE INDEX TO CONTROL WORDS (LS 40-4F)
:2823      SHIFT.OS(3-0)=79 ; 16 LOCATION HARDWARE INDEX TO SHIFT PATTERN
:2824                          ; (LS 20-2F)
:2825      SHIFT.OS(4-0)=7B ; 32 LOCATION HARDWARE INDEX TO SHIFT PATTERN
:2826                          ; (LS 20-3F)
:2827      OS=7C            ; OS REGISTER
:2828      DEC.CON=7D       ; DECIMAL CARRIES
:2829      SIZE=7E         ; SIZE REGISTER
:2830      MDT= 7E         ; MEMORY REFERENCE DATA SIZE
:2831      INTERRUPT.VEC=7E ; HARDWARE INTERRUPT VECTOR
:2832
:2833      ;
:2834      ; THIS WORD IS THE HARDWARE CONDITION CODES. THE CC'S CAN BE READ
:2835      ; OR ARE WRITTEN HERE. NO OTHER BITS IN THE PSL ARE EFFECTED.
:2836      PSL.CC=7F        ; PSL CONDITION CODES
```



```

:2837 .PAGE
:2838 XD.ADRS/=<16:9>,.VALIDITY=<XD.VAL>
:2839
:2840
:2841 SAV.WR0=1 ; STORAGE FOR WR0
:2842 SAV.WR1=2 ; STORAGE FOR WR1
:2843 SAV.WR2=3 ; STORAGE FOR WR2
:2844 SAV.WR3=4 ; STORAGE FOR WR3
:2845 T5.SUB=5 ; RESERVED FOR COMMON SUBROUTINES
:2846 T6.SUB=6 ; RESERVED FOR COMMON SUBROUTINES
:2847 T7.SUB=7 ; RESERVED FOR COMMON SUBROUTINES
:2848 TRANSFER.POINTER=7 ; POINTER FOR TRANSFER ROUTINE
:2849 MEMORY.SIZE=7 ; STORAGE FOR MEMORY SIZE IN 256KB BLOCKS AFTER
:2850 ; SIZING
:2851 FPA.FOUND=7 ; STORAGE FOR RESULT OF SIZING FOR FPA PRESENT
:2852 UBE.FOUND=7 ; STORAGE FOR RESULT OF SIZING FOR UBE PRESENT
:2853 T8=8 ; TEMP LOCATION 8
:2854 T9=9 ; TEMP LOCATION 9
:2855 T10=0A ; TEMP LOCATION 10
:2856 T11=0B ; TEMP LOCATION 11
:2857 T12=0C ; TEMP LOCATION 12
:2858 T13=0D ; TEMP LOCATION 13
:2859 T14=0E ; TEMP LOCATION 14
:2860 T15=0F ; TEMP LOCATION 15
:2861 PC=10 ; MACRO PROGRAM COUNTER
:2862
:2863 WR.BACKUP=11 ; BACKUP WR FOR MOV MEM.DATA TO WR[]
:2864 MM.LASTADDR+1=12 ; STORAGE OF LAST ADDRESS IN MAIN MEMORY PLUS
:2865 ; 1 (FIRST NON-EXISTANT MEMORY ADDRESS)
:2866 #FF=13 ; MASK
:2867 #FFFF=14 ; MASK
:2868 #FF000000=15 ; MASK
:2869 #FFFFFF00=16 ; MASK
:2870 CSR0.MASK=16 ; MASK
:2871 CSR1.MASK=17 ; MASK (01003FFF)
:2872 CSR2.MASK=18 ; MASK (FFFE3FFF)
:2873 #FE7FFFFFFF=19 ; MASK
:2874 UBS.SET=1A ; CONSTANT (7FF80000) USED BY UBS ADDRESS TEST
:2875 UBS.DATA=1B ; MASK (7FFF8000) USED AS ERROR MASK FOR UBS
:2876 ; DATA TEST
:2877
:2878 ERR.CON.NA=1E ; CONSTANT OF 800500(H) BITS 23,10,8 SET FOR
:2879 ; LOADING CONTROL AND STATUS REG IN INITIAL
:2880 ; TESTS
:2881 ERR.CON=1F ; CONSTANT OF 500(H) BITS 10 & 8 SET FOR
:2882 ; LOADING CONTROL AND STATUS REG IN INITIAL
:2883 ; TESTS
:2884
:2885 #1=20 ; PATTERN 00000001 (HEX)
:2886 #2=21 ; PATTERN 00000002
:2887 #4=22 ; PATTERN 00000004
:2888 #8=23 ; PATTERN 00000008
:2889 #10=24 ; PATTERN 00000010
:2890 #20=25 ; PATTERN 00000020
:2891 #40=26 ; PATTERN 00000040

```

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) 3-MAR-82 10:02:46
DIAGNOSTIC MACROS AND DEFINITIONS

DIAGNOSTIC MACROS A 3-MAR-82

Fiche 1 Frame H6

Sequence 72

Rev 01.00

Page 66

:2892	#80=27	: PATTERN 0C000080
:2893	#100=28	: PATTERN 00000100
:2894	#200=29	: PATTERN 00000200
:2895	#400=2A	: PATTERN 00000400
:2896	#800=2B	: PATTERN 00000800
:2897	#1000=2C	: PATTERN 00001000
:2898	#2000=2D	: PATTERN 00002000
:2899	#4000=2E	: PATTERN 00004000
:2900	#8000=2F	: PATTERN 00008000
:2901	#10000=30	: PATTERN 00010000
:2902	#20000=31	: PATTERN 00020000
:2903	#40000=32	: PATTERN 00040000
:2904	#80000=33	: PATTERN 00080000
:2905	#100000=34	: PATTERN 00100000
:2906	#200000=35	: PATTERN 00200000
:2907	#400000=36	: PATTERN 00400000
:2908	#800000=37	: PATTERN 00800000
:2909	#1000000=38	: PATTERN 01000000
:2910	#2000000=39	: PATTERN 02000000
:2911	#4000000=3A	: PATTERN 04000000
:2912	#8000000=3B	: PATTERN 08000000
:2913	#10000000=3C	: PATTERN 10000000
:2914	#20000000=3D	: PATTERN 20000000
:2915	#40000000=3E	: PATTERN 40000000
:2916	#80000000=3F	: PATTERN 80000000
:2917		
:2918	BIT0=20	: PATTERN 00000001
:2919	BIT1=21	: PATTERN 00000002
:2920	BIT2=22	: PATTERN 00000004
:2921	BIT3=23	: PATTERN 00000008
:2922	BIT4=24	: PATTERN 00000010
:2923	BIT5=25	: PATTERN 00000020
:2924	BIT6=26	: PATTERN 00000040
:2925	BIT7=27	: PATTERN 00000080
:2926	BIT8=28	: PATTERN 00000100
:2927	BIT9=29	: PATTERN 00000200
:2928	BIT10=2A	: PATTERN 00000400
:2929	BIT11=2B	: PATTERN 00000800
:2930	BIT12=2C	: PATTERN 00001000
:2931	BIT13=2D	: PATTERN 00002000
:2932	BIT14=2E	: PATTERN 00004000
:2933	BIT15=2F	: PATTERN 00008000
:2934	BIT16=30	: PATTERN 00010000
:2935	BIT17=31	: PATTERN 00020000
:2936	BIT18=32	: PATTERN 00040000
:2937	BIT19=33	: PATTERN 00080000
:2938	BIT20=34	: PATTERN 00100000
:2939	BIT21=35	: PATTERN 00200000
:2940	BIT22=36	: PATTERN 00400000
:2941	BIT23=37	: PATTERN 00800000
:2942	BIT24=38	: PATTERN 01000000
:2943	BIT25=39	: PATTERN 02000000
:2944	BIT26=3A	: PATTERN 04000000
:2945	BIT27=3B	: PATTERN 08000000
:2946	BIT28=3C	: PATTERN 10000000

```

2947      BIT29=3D      : PATTERN 2C000000
2948      BIT30=3E      : PATTERN 40000000
2949      BIT31=3F      : PATTERN 80000000
2950
2951
2952      ;CSR ADDRESS MNEMONICS
2953
2954      CSR0=4E      : ALL BITS CLEAR TO ACCESS CSR 0
2955      CSR1=22      : BIT 2 SET TO ACCESS CSR 1
2956      CSR2=23      : BIT 3 SET TO ACCESS CSR 2
2957
2958
2959      ;CONTROL AND STATUS WORD BITS
2960
2961      PRINT.UBE=26   : PRINT IF UBE PRESENT OR NOT PRESENT (BIT 6)
2962                    : (INDICATOR IN LS 7)
2963      PRINT.R80=27   : PRINT IF R80 PRESENT OR NOT PRESENT AND ON
2964                    : WHICH DRIVE (BIT 7). INDICATOR IN LS 7,
2965                    : DRIVE NUMBER IN LS 6.
2966      ERROR=28       : TEST ERROR INDICATION (BIT 8)
2967      SET.PA.ERR=29   : TELLS CONSOLE TO CREATE A PARITY ERROR AT WCS
2968                    : ADDRESS POINTED TO BY LS 7 (BIT 9)
2969      CONSOLE.LOE=2A  : INDICATES CONSOLE DOES ERROR LOOPING (BIT 10)
2970                    : (FOR INITIAL TESTS)
2971      PRINT.MEM.SIZE=2B : PRINT MEMORY SIZE IN 256K BLOCKS (BIT 11)
2972                    : (SIZE IN LS 7)
2973      PRINT.FPA=2C    : PRINT IF FPA PRESENT OR NOT PRESENT (BIT 12)
2974                    : (INDICATOR IN LS 7)
2975      XFER.DATA=2D    : INDICATES A DATA TRANSFER TO LS OR MM (BIT 13)
2976      EOS=2E         : END OF SECTION (BIT 14)
2977      BEGIN.TEST=2F   : BEGINNING OF TEST (BIT 15)
2978      INTERRUPT.EN=30 : ENABLE INTERRUPT OR SIGNAL SPECIFIED BY BITS
2979                    : 17 THRU 22 AND 28 THRU 30 (BIT 16)
2980
2981      CONSOLE.HALT=31 : CONSOLE HALT INTERRUPT (BIT 17)
2982      PWR.FAIL=32     : PWR FAIL INT (BIT 18)
2983      INTERVAL.TIM=33 : INTRVL TIM INT (BIT 19)
2984      CONSOLE.ATTN=34 : CONSOLE ATTN INTERRUPT (BIT 20)
2985      CONSOLE.ACK=35   : CONSOLE ACK INTERRUPT (BIT 21)
2986      DISABLE.MEM.REF=36 : DISABLE MEMORY REFERENCES (BIT 22)
2987      CINIT=3C         : UNIBUS INIT SIGNAL TO THE MCT (BIT 28)
2988      UBS.DCLO=3D      : UNIBUS DC LO INDICATION TO THE MCT (BIT 29)
2989      UBS.BBSY=3E      : UNIBUS BUS BUSY INDICATOR TO MCT (BIT 30)
2990
2991      NA.EXP.REC=37    : PRINT N/A UNDER EXPECTED AND RECEIVED (BIT 23)
2992      OTHER.DATA=38    : PRINT A VALUE UNDER OTHER DATA (BIT 24)
2993      CONSOLE.LOOP=39  : CONSOLE PERFORMS A LOOP ACCORDING TO LOOP
2994                    : COUNT IN LS (BIT 25)
2995      PRINT.CRD=3A     : PRINT NUMBER OF SINGLE BIT ERRORS (BIT 26)
2996      CLR.PA.ERR=3B    : TELLS CONSOLE TO CLEAR PARITY ERROR AT WCS
2997                    : ADDRESS POINTED TO BY LS 7 (BIT 27)
2998      PRINT.IDC=3F     : PRINT IF IDC PRESENT OR NOT PRESENT (BIT 31)
2999                    : (INDICATOR IN LS 7)
3000
3001

```

ZZ-ENKCB-1.0
 : ENKCB.MCR
 : ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) 3-MAR-82 10:02:46
 DIAGNOSTIC MACROS AND DEFINITIONS

DIAGNOSTIC MACROS A J 6

Fiche 1 Frame J6

Sequence 74

Rev 01.00

Page 68

ENKCB Micro-diagnostic for DAP module

```

:3002 ;MODULE CODES
:3003
:3004     WCS=20      ; MODULE CODE FOR WCS BOARD (BIT 0 SET)
:3005     CPU=21     ; MODULE CODE FOR CPU BOARD (BIT 1 SET)
:3006     MCT=22     ; MODULE CODE FOR MCT BOARD (BIT 2 SET)
:3007     FPA=23     ; MODULE CODE FOR FPA BOARD (BIT 3 SET)
:3008     ARRAY=24   ; MODULE CODE FOR ARRAY BOARD (BIT 4 SET)
:3009     IDC=25     ; MODULE CODE FOR IDC BOARD (BIT 5 SET)
:3010
:3011 ;CSR 1 ERROR BITS
:3012
:3013     VALID.ERR=2E ; VALID NOT SET IF 1 (BIT 14)
:3014     TB.PAR.ERR=2F ; TB PARITY ERROR (BIT 15)
:3015     NXM=30      ; NON-EXISTANT MEMORY ERROR (BIT 16)
:3016     UB.BSY=31   ; UNIBUS BUSY (BIT 17)
:3017     ADP.REG=32   ; UNIBUS ADAPTER REGISTER SELECT (BIT 18)
:3018     WR.ACROSS.PG=33 ; WRITE ACROSS PAGE BOUNDARY ERROR (BIT 19)
:3019     OP.ERR=34    ; OPERATION ERROR (BIT 20)
:3020     TB.MISS=35   ; TB MISS (VIRTUAL ADDRESS TRANSLATION NOT IN
:3021                   ; BUFFER) (BIT 21)
:3022     ACCESS.REFUSED=36 ; PROTECTION ERROR ON ACCESS (BIT 22)
:3023     MODIFY.REFUSED=37 ; PROTECTION ERROR ON WRITE (BIT 23)
:3024
:3025     CRD=3E       ; SINGLE BIT ERROR CORRECTED (BIT 30)
:3026     RDS=3F       ; UNCORRECTABLE DATA ERROR (BIT 31)
:3027
:3028
:3029 ;CSR 1 CONTROL BITS
:3030
:3031     ECC.DIS=39    ; CSR1 ECC DISABLE BIT (BIT 25)
:3032     DIAG.CHK=3A   ; CSR1 DIAG CHK BIT (BIT 26)
:3033     MME=3B        ; CSR1 MEMORY MANAGMENT ENABLE BIT (BIT 27)
:3034     INH.CRD=3C    ; CSR1 INHIBIT REPORT CRD BIT (BIT 28)
:3035     TB.PAR.DIAG=3D ; CSR1 TB PAR DIAG BIT (FORCE TB PARITY ERR)
:3036                   ; (BIT 29)
:3037
:3038
:3039 ;UBE CONTROL REGISTER BITS
:3040
:3041 ;CONTROL REG 1
:3042     GO=20         ; START UBE (BIT 0)
:3043     BR4=21        ; ISSUE BUS REQUEST 4 (BIT 1)
:3044     BR5=22        ; ISSUE BUS REQUEST 5 (BIT 2)
:3045     BR6=23        ; ISSUE BUS REQUEST 6 (BIT 3)
:3046     BR7=24        ; ISSUE BUS REQUEST 7 (BIT 4)
:3047     NPR=25        ; ISSUE NPR (BIT 5)
:3048     INT.ODNE=26   ; INTERRUPT ON DONE (WHEN CCOVF) (BIT 6)
:3049     RDY=27        ; READY BIT, SET WHEN READY TO BEGIN FUNC (BIT 7)
:3050     C0=28         ; C0 BIT (BIT 8)
:3051     C1=29         ; C1 BIT (BIT 9)
:3052     FUNA=2A       ; FUNCTION BIT A (BIT 10)
:3053     FUNB=2B       ; FUNCTION BIT B (BIT 11)
:3054     CC+DAT=2C      ; DATA FROM CYCLE COUNT IF SET, DATA REG IF CLEAR
:3055     INH.DATIP.ROL=2D ; INHIBIT ROL ON DATIP (BIT 13)
:3056     DATOB.ON.DATIP=2E ; DO DATOB AFTER DATIP CYCLE (BIT 14)

```

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

DIAGNOSTIC MACROS A 3-MAR-82
MICRO2 1L(03) 3-MAR-82 10:02:46
DIAGNOSTIC MACROS AND DEFINITIONS

K 6

Fiche 1 Frame K6

Sequence 75

Rev 01.00

Page 69

```
:3057      ERR=2F      : EXERCISOR OR UNIBUS ERROR (BIT 15)
:3058
:3059      :CONTROL REG 2
:3060      EXT.MEM0=20      : EXTENDED MEMORY BIT 0 (BIT 0)
:3061      EXT.MEM1=21      : EXTENDED MEMORY BIT 1 (BIT 1)
:3062      INH.INC.ADDR&CC=22      : INHIBIT INC ON CYCLE COUNT AND ADDR REG (BIT 2)
:3063      INH.SACK=23      : INHIBIT ASSERTING BUS SACK ON BUS GRANT (BIT 3)
:3064      PWR.DWN.SEQ=24      : ASSERT ACLO (BIT 4)
:3065      WNG.GNT.BCK=25      : NO BUS GRANT RECV TO HIGEST REQUEST (BIT 5)
:3066      MAX.LATE.ERR=26      : NPR AND BR LATENCY TIME EXCEEDED (BIT 6)
:3067      NO.SACK.ERR=27      : NO TIMEOUT AT CPU WHEN NO SACK (BIT 7)
:3068      NO.SSYN.ERR=28      : NO SSYN RECEIVED AFTER ASSERTING MSYN (BIT 8)
:3069      WRG.A.LINES=29      : ADDRESS ERR ON SENT & RECEIVED ADDRESS (BIT 9)
:3070      NO.GNT+NOT.ONE.GNT=2A      : NO GRANT OR MORE THAN 1 GRANT AFTER REQ. (BIT 10)
:3071      INTR.SSYN.ERR=2B      : NO SSYN RECEIVED AFTER BUS INTR (BIT 11)
:3072      ENB.PB=2C      : ENABLE PARITY BIT B (BIT 12)
:3073      CCOVF=2D      : CYCLE COUNT OVERFLOW (BIT 13)
:3074      TM.DLY=2E      : REQUEST BUS EVERY 10 USEC (BIT 14)
:3075
:3076      :BG6 MNEMONIC
:3077      BG6=23      : BIT 3 SET FOR BG 6 ADDRESS
:3078
:3079      :ERROR AND CONTROL INFORMATION
:3080
:3081      CONTROL.STATUS=40      : CONTROL AND STATUS WORD
:3082      ERROR.NUMBER=41      : ERROR NUMBER OF CURRENT TEST
:3083      EXPECTED.DATA=42      : EXPECTED RESULT OF CURRENT TEST
:3084      RECEIVED.DATA=43      : ACTUAL RESULT OF CURRENT TEST
:3085      ADDRESS.DATA=44      : OTHER PERTINENT DATA
:3086      ERROR.MASK=45      : BITS TO CHECK IN RESULT
:3087      MODULE.NUM=46      : STORAGE OF MODULE NUMBER (CODED)
:3088      LOOP.CNT=47      : STORAGE OF LOOP COUNT FOR CONSOLE LOOP
:3089      CONTROL
:3090      ERROR.CONTROL=48      : STORAGE FOR ERROR CONTROL (LOOP ON ERROR ETC.)
:3091      LOE=20      : LOOP ON ERROR IF SET (BIT0)
:3092      NER=21      : NO ERROR REPORT IF SET (BIT1)
:3093      BELL=22      : BELL ON ERROR IF SET (BIT2)
:3094      HOE=23      : HALT ON ERROR IF SET (BIT3)
:3095      SER=24      : ENABLE ERROR REPORT ON CRD ERRORS IF SET
:3096      APT.PRESENT=25      : APT PRESENT IF SET (BIT5)
:3097      LOOP.COMMAND=26      : LOOP COMMAND TYPED IF SET (BIT 6)
:3098
:3099      APT.PARAM=49      : APT RUN TIME PARAMETER REGS (MEM SIZE, HARD-
:3100      : WARE CONFIG, SOFTWARE CONFIG IN BYTES 0, 1
:3101      : AND 2 RESPECTIVELY. BYTE 3 FOR FUTURE USE)
:3102      APT.RESERVED=4A      : RESERVED FOR FUTURE APT USE
:3103
:3104      :MISCELLANEOUS CONSTANTS AND STORAGE
:3105
:3106      #AAAAAAA=4C      : CONSTANT
:3107      ALTER.ODD=4C      : CONSTANT
:3108      #55555555=4D      : CONSTANT
:3109      ALTER.EVEN=4D      : CONSTANT
:3110      #0=4E      : CONSTANT
:3111      ZERO=4E      : CONSTANT
```

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

DIAGNOSTIC MACROS A 3-MAR-82 L 6
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module
DIAGNOSTIC MACROS AND DEFINITIONS

Fiche 1 Frame L6

Sequence 76

Rev 01.00

Page 70

```
:3112      #FFFFFFF=4F      : CONSTANT
:3113      ONES=4F         : CONSTANT
:3114      PREVIOUS.ERROR=50 : ERROR NUMBER OF LAST ERROR
:3115
:3116      DATA.1=51       : STORAGE FOR FPA DATA
:3117      DATA.2=52       : STORAGE FOR FPA DATA
:3118      DATA.3=53       : STORAGE FOR FPA DATA
:3119      FIRST.FLT=54     : STORAGE FOR FPA DATA
:3120      SEC.FLT=55       : STORAGE FOR FPA DATA
:3121      THIRD.FLT=56    : STORAGE FOR FPA DATA
:3122      T57=57          : TEMP LOCATION 57
:3123      T58=58          : TEMP LOCATION 58
:3124      T59=59          : TEMP LOCATION 59
:3125
:3126      BEDB=5A         :UBE (BUS EXERCISOR) DATA BUF REG
:3127      BECC=5B         :UBE (BUS EXERCISOR) CYCLE COUNT REG
:3128      BEBA=5C         :UBE (BUS EXERCISOR) ADDR REG
:3129      BECR1=5D        :UBE (BUS EXERCISOR) CONTROL REG 1
:3130      CLEAR.ERR.ADDR=5E :UBE CLEAR ERROR REG ADDRESS
:3131      BECR2=5F        :UBE (BUS EXERCISOR) CONTROL REG 2
:3132
:3133      #3(H)=60        : CONSTANT
:3134      #12(H)=61       : CONSTANT
:3135      #33333333=62    : CONSTANT
:3136      #0F0F0F0F=63    : CONSTANT
:3137      #00FF00FF=64    : CONSTANT
:3138      #162(H)=65      : CONSTANT FOR LS TRANSFER POINTER
:3139      BR7.IDENT=66    : BR7 IDENTIFIER (1010(B) IN BITS 5-2)
:3140      BG7=67         : BG7 ADDRESS (BITS 2 AND 3 SET)
:3141
:3142      ;TEMPS FOR CPU TESTS
:3143      L30=30          :LS 30 TEMP (RESTORED TO 10000 AFTER USE)
:3144      L31=31          :LS 31 TEMP (RESTORE TO 20000 AFTER USE)
:3145      L53=53          :LS 53 TEMP
:3146      L73=73          :LS 73 TEMP
:3147
:3148      ;REGISTER ADDRESSES
:3149
:3150      CONTROL.OS=78    : HARDWARE INDEX TO CONTROL WORDS (LS 40-4F)
:3151      SHIFT.OS(3-0)=79 : 16 LOCATION HARDWARE INDEX TO SHIFT PATTERN
:3152                        : (LS 20-2F)
:3153      SHIFT.OS(4-0)=7B : 32 LOCATION HARDWARE INDEX TO SHIFT PATTERN
:3154                        : (LS 20-3F)
:3155      OS=7C            : OS REGISTER
:3156      DEC.CON=7D       : DECIMAL CARRIES
:3157      SIZE=7E          : SIZE REGISTER
:3158      MDT= 7E         : MEMORY REFERENCE DATA SIZE
:3159      INTERRUPT.VEC=7E : HARDWARE INTERRUPT VECTOR
:3160
:3161      :
:3162      : THIS WORD IS THE HARDWARE CONDITION CODES. THE CC'S CAN BE READ
:3163      : OR ARE WRITTEN HERE. NO OTHER BITS IN THE PSL ARE EFFECTED.
:3164      :
:3165      PSL.CC=7F         : PSL CONDITION CODES
:3166
```

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) COMMON LS ASSIGNMEN 3-MAR-82
COMMON LS ASSIGNMENTS (TO REGISTERS)

6

Fiche 1 Frame M6

Sequence 77

ENKCB Micro-diagnostic for DAP module Rev 01.00 Page 71

```
:3167 .TOC "COMMON LS ASSIGNMENTS (TO REGISTERS)"
:3168
:3169 .UPPER HALF OF LS
:3170
:3171 XGPR.OS=0F8 : HARDWARE INDEX TO XGPRS
:3172 USER.INDEX(3-0)=0F9 : 16 LOCATION HARDWARE INDEX TO DIAGNOSTIC DATA
:3173 : AREA (LS LOCATIONS 0A0 THROUGH 0AF)
:3174 USER.INDEX(4-0)=0FB : 32 LOCATION HARDWARE INDEX TO DIAGNOSTIC DATA
:3175 : AREA (LS LOCATIONS 0A0 THROUGH 0BF)
:3176 CCR=0FC : CONSOLE READ REGISTER
:3177 TIMER=0FD : INTERVAL TIMER VALUE.
:3178 ALU.CC=0FE : ALU CONDITION CODES
:3179
:3180 THIS LOCATION IS A WRITE ONLY REGISTER. THE FOLLOWING BITS IN THE
:3181 PSL ARE AFFECTED:
:3182
:3183 COMPATIBILITY MODE - BIT<31>
:3184 CURRENT MODE - BITS<25:24>
:3185 INTERRUPT PRIORITY LEVEL (IPL) - BITS<20:16>
:3186 TRACE TRAP ENABLE (REALLY T OR TP) - BIT<4>
:3187 NEGATIVE CONDITION CODE - BIT<3>
:3188 ZERO CONDITION CODE - BIT<2>
:3189 OVERFLOW CONDITION CODE - BIT<1>
:3190 CARRY CONDITION CODE - BIT<0>
:3191
:3192 PSL.HW=OFF : HARDWARE PSL BITS.
:3193 .TOC "PROGRAM SPECIFIC LS ASSIGNMENTS (LS 80-F7)"
:3194
:3195 THESE ADDRESSES ARE ONLY ACCESSIBLE WITH THE MOV MICRO INSTRUCTION.
:3196 STARTING AT ADDRESS 80(H). THEY ARE NOT USED BY ALL SOFTWARE MODULES
:3197 BUT ARE SPECIFIC TO THIS ONE.
:3198
:3199
:3200
:3201 L90=90 :TEMP LOCATION
:3202 ALTER.MIX=91 :DATA: AAAA5555
:3203 HI.IPL=92 :BITS 16-20 SET (IPL=1F)
:3204 #FFFFFFFC=95 :FREQUENTLY USED MASK (BOTTOM BYTE)
:3205 #000F=96
:3206 #00F0=97
:3207 #0F00=98
:3208 #F000=99
:3209 #30000=9A
:3210 #7FFF8000=9B
:3211 #540000=9C
:3212 #F00000=9D
:3213 #FC0000=9E
:3214 #3F003FFF=9F
:3215 #254=0A0 :POINTER TO MEMORY XFER DATA
:3216
:3217 LB0=0B0 :TEMP LOCATION
:3218
:3219 LD0=0D0 :TEMP LOCATION
:3220
:3221 LF0=0F0 :TEMP LOCATION
```

3222 page 'Microinstruction Formats'

3223
3224 The following is taken directly from the NEBULA Hardware
3225 Specification.

MICROINSTRUCTION FORMATS

3226		22 21 16 15 9 8 7 6 5 4 0
3227		+-----+-----+-----+-----+-----+
3228	1. BASIC	1 DP D ADRS B CC SCTL
3229		+-----+-----+-----+-----+-----+
3230		22 20 19 17 16 9 8 7 6 5 4 0
3231		+-----+-----+-----+-----+-----+
3232	2. MOVE	0 1 1 MDP XD ADRS B CC SCTL
3233		+-----+-----+-----+-----+-----+
3234		22 20 19 14 13 11 9 8 7 6 5 4 0
3235		+-----+-----+-----+-----+-----+
3236	3. EXTENDED	0 1 0 XDP SI A B CC SCTL
3237		+-----+-----+-----+-----+-----+
3238		22 19 18 16 15 9 8 7 6 5 4 0
3239		+-----+-----+-----+-----+-----+
3240	4. MEM.REQ	0 0 1 1 MF1 D ADRS MF2 DT SCTL
3241		+-----+-----+-----+-----+-----+
3242		22 19 18 16 15 12 11 9 8 7 6 5 4 0
3243		+-----+-----+-----+-----+-----+
3244	5. MISC	0 0 1 0 P 0 0 M1 M2 0 R 0 0 SCTL
3245		+-----+-----+-----+-----+-----+
3246		22 19 18 17 4 3 0
3247		+-----+-----+-----+-----+-----+
3248	6. JUMP	0 0 0 1 OS JUMP ADDRESS JCTL
3249		+-----+-----+-----+-----+-----+
3250		22 19 18 17 12 11 9 8 7 6 5 4 3 0
3251		+-----+-----+-----+-----+-----+
3252	7. DECODE	0 0 0 0 DEC.ADRS IFUNC JCTL
3253		+-----+-----+-----+-----+-----+
3254		BACK UP PC IB REQ---+---OPC/SPEC
3255		SEL CM HI IR BYTE---+---LOAD RDEST
3256		LOAD OS---

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

Microinstruction Fo 3-MAR-82
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module
Microinstruction Formats

Fiche 1 Frame B7

Sequence 79

Rev 01.00

Page 73

.3272 .page
.3273 .
.3274 .
.3275 1. 'BASIC' Microinstruction
.3276 .
.3277 CSR<22> = 1 (OPCODE)
.3278 .
.3279 This opcode bit causes LS Adrs <7> to be cleared.
.3280 .
.3281 CSR<21:16> (Data Path Control)
.3282 .
.3283 This field controls the data path. It controls the 2901 ALU,
.3284 the ALU data source, the data destination in the 2901 array,
.3285 and the write to the Local Store.
.3286 .
.3287 CSR<15:9> (D Adrs <6:0>)
.3288 .
.3289 This field selects which of the 128 accessible Local Store
.3290 locations to use.
.3291 .
.3292 CSR<8:7> (B Adrs)
.3293 .
.3294 This field selects one of the four Working Registers.
.3295 .
.3296 CSR<6:5> (Condition Codes)
.3297 .
.3298 This field specifies the data type and condition code control.
.3299 .
.3300 CSR<4:0> (SCTL)
.3301 .
.3302 This field specifies the skip condition/micro sequencer
.3303 function.

:3304
:3305
:3306
:3307
:3308
:3309
:3310
:3311
:3312
:3313
:3314
:3315
:3316
:3317
:3318
:3319
:3320
:3321
:3322
:3323
:3324
:3325
:3326
:3327
:3328
:3329
:3330
:3331
:3332
:3333
:3334
:3335
:3336
:3337
:3338
:3339
:3340
:3341
:3342
:3343
:3344
:3345
:3346
:3347

.page

2. 'MOVE' Microinstruction

CSR<22:20> = 011 (OPCODE)

This combination of opcode bits allows CSR<16> to control LS Adrs <7>. This permits access of LS locations 80 - FF.

CSR<19:17> (Data Path Control)

This field is decoded to produce some data path control information.

0 LS[XD] <- WR[B], WR[B] <- MEM DATA*	4 LS[XD] <- MEM DATA*
1 MEMORY <- LS [XD]*	5 LS[XD] <- ACC/PORT
2 Q <- XWR[B]	6 LS[XD] <- WR[B] + OS
3 WR[B] <- LS [XD]	7 LS[XD] <- WR[B]

CSR<16:9> (XD<7:0>)

This field specifies which of the 256 Local Store locations to use.

CSR<8:7> (B Adrs)

This field specifies which of the four Working Registers to use. For MDP = 2, this field selects either the Backup PC or the VAR.

CSR<6:5> (CC)

This field specifies the data type and condition code control.

CSR<4:0> (SCTL)

This field specifies the Skip control. See Table 3.

* For information on which Working Registers and Local Store locations may be used for Memory Addresses and data source/destinations, see the Memory Management documentation. Note that these restrictions are due to microcoding conventions and not hardware.

3348 .page
3349
3350
3351 3. 'EXTENDED' Microinstruction
3352
3353 CSR<22:20> = 010 (OPCODE)
3354
3355 This opcode causes a function internal to the 2901 array; that
3356 is, an operation involving only Working Registers and the Q-
3357 Register.
3358
3359 CSR<19:14> (Data Path Control)
3360
3361 This field is decoded to supply the 2901 ALU function code,
3362 the ALU Source control, the destination code, and the ALU
3363 Carry-In.
3364
3365 CSR<13:11> (Shift Input)
3366
3367 This field selects the data to be shifted into a register,
3368 when a Shift function is being done.
3369
3370 CSR<10:9> (A Adrs)
3371
3372 This field selects which Working Register to use as a source
3373 of data, when RAM A is selected to the ALU.
3374
3375 CSR<8:7> (B Adrs)
3376
3377 This field selects which Working Register to use as a source
3378 (when RAM B is selected to the ALU) and/or destination (when
3379 the RAM is written) of data.
3380
3381 CSR<6:5> (CC)
3382
3383 This field specifies the data type and condition code control.
3384
3385 CSR<4:0> (SCTL)
3386
3387 This field specifies the skip condition/micro sequencer
3388 function.

22-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) Microinstruction Fo 3-MAR-82
3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module
Microinstruction Formats

Fiche 1 Frame E7

Sequence 82
Rev 01.00

Page 76

:3389 .page
:3390
:3391
:3392 4. 'MEM.REQ' Microinstruction
:3393
:3394 CSR<22:19> = 0011 (OPCODE)
:3395
:3396 This opcode causes a Memory Request to be started. The Local
:3397 Store is output on on the D-Bus, to be used as the virtual
:3398 address. Working Register 5 is written with this address.
:3399
:3400 CSR<18:16>, <8:7> (Memory Function)
:3401
:3402 This field specifies to the Memory Controller the type of
:3403 request: physical, virtual, type of access, etc. For the
:3404 list of functions and the codes, see the NÉBULA Memory Spec.
:3405
:3406 CSR<15:9> (D Adrs <6:0>)
:3407
:3408 This field selects which of the 128 accessible Local Store
:3409 locations to use as the virtual address.
:3410
:3411 CSR<6:5> (Data Type)
:3412
:3413 This field specifies the data type of the request.
:3414
:3415 00:=BYTE
:3416 01:=WORD
:3417 10:=SIZE Reg
:3418 11:=LONG
:3419
:3420 CSR<4:0> (SCTL)
:3421
:3422 This field specifies the skip condition/micro sequencer
:3423 function.

:3424
:3425
:3426
:3427
:3428
:3429
:3430
:3431
:3432
:3433
:3434
:3435
:3436
:3437
:3438
:3439
:3440
:3441
:3442
:3443
:3444
:3445
:3446
:3447
:3448
:3449
:3450
:3451
:3452
:3453
:3454
:3455
:3456
:3457
:3458
:3459
:3460
:3461
:3462
:3463
:3464
:3465
:3466
:3467
:3468
:3469
:3470
:3471
:3472
:3473
:3474
:3475
:3476

.page

5. 'MISC' Microinstruction

CSR<22:19> = 0010 (OPCODE)

This combination of opcode bits causes the data path to no-op,
and enables the MISC decoders.

CSR<18> (Port/Misc Select)

This bit defines the instruction to be either a Misc or a Port
instruction. For CSR<18> = 1, CSR<17:10> is the Command Byte
to the Port, CSR<9:5> must be zero and CSR<4:0> is the STL
field. For CSR<18> = 0, it is a Misc instruction and the rest
of the word is defined as follows.

CSR<17:16> (Not Used)

CSR<15:12> (Misc Function 1)

These four bits are decoded to do the following functions:

- 0 = CLEAR State Reg<1:0>
- 1 = CLEAR State Reg<1>, SET State Reg<0>
- 2 = SET State Reg<1>, CLEAR State Reg<0>
- 3 = CLEAR State Reg<0>
- 4 = CLEAR State Reg<1>
- 5 = SET State Reg<0>
- 6 = SET State Reg<1>
- 7 = SET REGISTER BACKUP MASK FLAG
- 8 = SET ACC I/O MODE
- 9 = SET PORT I/O MODE
- A = CLEAR WCS HI ADRS
- B = SET WCS HI ADRS
- C = CLEAR CPU ATTN AND CPU ACK
- D = SET CPU ACK
- E = SET CPU ATTN
- F = NOP

CSR<11:9> (Misc Function 2)

This field is decoded to do the following functions:

- 0 = NOP
- 1 = Mask IRD Interrupt Detection during Next State
- 2 = Assert CPU Data Available and pass WR to ACC/Port
- 3 = Trap ACC
- 4 = Read ACC uPC
- 5 = Assert XFER GRANT to Port
- 6 = Mask Halt and I-Bit Traps
- 7 = Write WR to Console Read Register

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) Microinstruction Fo 3-MAR-82
Microinstruction Formats

Fiche 1 Frame G7

Sequence 84

Page 78

:3477 :.page
:3478 :
:3479 : CSR<8> (MUST BE ZERO)
:3480 :
:3481 : CSR<7> (WR Address)
:3482 :
:3483 :
:3484 : This bit selects WR[0] or WR[1] to be put on the Y-Bus.
:3485 :
:3486 : CSR<6:5> (Not Used)
:3487 :
:3488 : CSR<4:0> (SCTL)
:3489 :
:3490 : This field specifies the skip condition/micro sequencer
:3491 : function.
:3492 :

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) Microinstruction Fo 3-MAR-82
Microinstruction Formats

Fiche 1 Frame H7

Sequence 85

Rev 01.00

Page 79

:3493 :page
:3494 :
:3495 :
:3496 :6. "JUMP" Microinstruction
:3497 :
:3498 :CSR<22:19> = 0001 (OPCODE)
:3499 :
:3500 :This opcode causes the data path to no-op, and the micro
:3501 :sequencer to execute a JUMP.
:3502 :
:3503 :CSR<18> (Select OS)
:3504 :
:3505 :This bit, when asserted, causes OS<4:0> to be OR'ed with the
:3506 :five low bits of the of the jump microaddress.
:3507 :
:3508 :CSR<17:4> (Jump Microaddress)
:3509 :
:3510 :If Select OS is CLEAR, this field specifies the fourteen-bit
:3511 :jump micro address. If Select OS is SET, CSR<17:9> is Jump
:3512 :Adrs<13:5> and OS<4:0> OR'ed with CSR<8:4> is Jump Adrs<4:0>.
:3513 :
:3514 :CSR<3:0> (JCTL)
:3515 :
:3516 :This field specifies the jump condition/micro sequencer
:3517 :function.

:3518 .page
:3519
:3520
:3521 7. "DECODE" Microinstruction
:3522
:3523 A. DESCRIPTION
:3524
:3525 (CSR<22:19> = 0000 (OPCODE)
:3526
:3527 This combination of opcode bits causes the Local Store to
:3528 access location 10 (the PC.) The Local Store drives the D-Bus.
:3529 The 2901 is set up to input the data on the D-Bus, increment
:3530 it, and output to the Y-Bus. Thus, the Y-Bus contains the PC
:3531 + 1.
:3532
:3533 (CSR<18> (BPC [asserted low])
:3534
:3535 If this bit is CLEAR, the incremented PC is written into the
:3536 2901 RAM. If it is SET, the RAM is not written.
:3537
:3538 (CSR<17:12> (DEC.ADRS <13:8>)
:3539
:3540 This data is taken to be the upper six bits of the next
:3541 microaddress. The low eight bits are supplied by the IRD ROM.
:3542 The actual JUMP/JSR/NO-OP is determined by the JCTL field.
:3543
:3544 (CSR<11:9> (IFUNC)
:3545
:3546 This is the IFUNC field. It defines which fork is being
:3547 taken.
:3548
:3549 (CSR<8> (IB.REQ)
:3550
:3551 This is the IB.REQ bit. When it is CLEAR, the LS is not
:3552 written, no interaction with memory is initiated and the
:3553 instruction buffer is not affected.
:3554
:3555 When it is SET:
:3556
:3557 The incremented PC (valid on the Y-Bus by the end of the
:3558 cycle) is re-written to Local Store location 0. This
:3559 completes the PC increment function.
:3560
:3561 A Conditional Memory Request is executed. If the two low bits
:3562 of the PC are not 11, the request is not initiated. If the
:3563 two low bits are set, an IB PREFETCH is done. The
:3564 (unincremented) PC is output to the memory and the CPU grant
:3565 (CPUG) signal is examined. If CPUG is low by T120, the CPU
:3566 will stall until the CPUG signal becomes true. After the
:3567 request is completed, the next state entered is dependent upon
:3568 the JCTL field.

3569 : page
 3570 :
 3571 : CSR<7> (SEL CM HI IR BYTE)
 3572 :
 3573 : This bit enables the upper byte of the Compatibility Mode
 3574 : instruction in the Prefetch register to drive the IB-Bus. It
 3575 : is used only while in Compatibility Mode, when IRD is being
 3576 : done. It must never be asserted in VAX Mode.
 3577 :
 3578 : CSR<6> (LD.OS)
 3579 :
 3580 : If this bit is SET, the eight-bit data on the IB Bus (the
 3581 : output of the instruction buffer) is loaded into the OS
 3582 : register. This load is dependent upon Compatibility Mode and
 3583 : LD R Dest (If CM.AND.LD R Dest, OS<3> <- 0.) If the bit is
 3584 : CLEAR, OS is not affected.
 3585 :
 3586 : CSR<5> (LD R DEST)
 3587 :
 3588 : If this bit is SET, the R Dest Skip bit will be SET for
 3589 : Specifier Mode equal to 5 and CLEARED for Mode not equal to 5,
 3590 : in VAX mode, or mode 0 while in Compatibility Mode. If this
 3591 : bit is CLEAR, the R Dest bit will be unaffected.
 3592 :
 3593 : CSR<4> (OPC/SPEC)
 3594 :
 3595 : This bit (effectively another IFUNC bit) defines the data to
 3596 : be decoded; that is, if it is an Opcode or a Specifier. In
 3597 : hardware, it selects which ROM is to drive the micro address
 3598 : bus.
 3599 :
 3600 : CSR<3:0> (JCTL)
 3601 :
 3602 : These four bits are decoded to make one of 16 functions. See
 3603 : Table 3.
 3604 :

```

:3605 page "Tables"
:3606
:3607 TABLES
:3608
:3609
:3610
:3611 Table 1. 2901 Control Decoding
:3612
:3613 The 2901 control decoder examines CSR<22:14>. This nine-bit
:3614 field of the microword changes in meaning for the different
:3615 cases of micro opcode.
:3616
:3617 CSR      22  21  20  19  18  17  16  15  14
:3618 --      --  --  --  --  --  --  --  --
:3619
:3620 1. BASIC      1 [Six bit function code] X  X
:3621 2. MOVE      0  1  1 [function] X  X  X
:3622 3. EXTENDED  0  1  0 [Six bit function code]
:3623 4. MEM.REQ   0  0  1  1  X  X  X  X  X
:3624 5. MISC     0  0  1  0  X  X  X  X  X
:3625 6. JUMP     0  0  0  1  X  X  X  X  X
:3626 7. DECODE   0  0  0  0  BPC* X  X  X  X
:3627
:3628 If BPC = 0, Write WR; else, No-op
:3629
:3630 This decoder is implemented using a ROM and a PAL. 512
:3631 locations are needed for CSR<22:14> to index into. The
:3632 decoding logic supplies 10 control bits:
:3633
:3634 ALU Source Select (3)
:3635 ALU Function (3)
:3636 ALU Result Destination (3)
:3637 ALU Carry-In (1)
:3638
:3639 The ROM addresses are therefore allocated in the following way:
:3640
:3641 Location      0 - F      Incr. D-Bus, Output ALU, No Write.
:3642             10 - 1F     Incr. D-Bus, Output ALU, Write RAM.
:3643             20 - 3F     No-Op.
:3644             40 - 5F     Output WR, bypassing ALU.
:3645             60 - 7F     Pass D-Bus through ALU and write RAM.
:3646             80 - BF     CSR<19:14> indicates 2901 function.
:3647             C0 - FF     CSR<19:17> indicates 2901 function.
:3648             100 - 1FF   CSR<21:16> indicates 2901 function.
:3649
:3650 For the detailed function table see the Microcode Listing.

```

```

WR[5] <- D
Y-Bus <- WR
NO-OP

```

:3651
:3652
:3653
:3654
:3655
:3656
:3657
:3658
:3659
:3660
:3661
:3662
:3663
:3664
:3665
:3666
:3667
:3668
:3669
:3670
:3671
:3672
:3673
:3674
:3675
:3676
:3677
:3678
:3679
:3680
:3681
:3682
:3683
:3684
:3685
:3686
:3687
:3688
:3689
:3690

:page

Table 2. Local Store Write

The Local Store Write Controller examines CSR<22:17>, CSR<6>, the SIZE register (SIZE<1:0>) and the Compatibility Mode bit (CM). These eight bits are decoded to determine which parts of the LS to write, if any. The LS is written with data from the 2901 ALU (Y-Bus).

CSR	22	21	20	19	18	17	
--	--	--	--	--	--	--	
1. BASIC	1	0	X	X	X	X	No Write
	1	1	X	X	X	X	Write *
2. MOVE	0	1	1	0	X	1	No Write
	0	1	1	0	1	X	No Write
	0	1	1	1	X	X	Write *
3. EXTENDED	0	1	1	X	X	X	No Write
4. MEM.REQ	0	0	1	1	X	X	No Write
5. MISC	0	0	1	0	X	X	No Write
6. JUMP	0	0	0	1	X	X	No Write
7. DECODE	0	0	0	0	X	X	Conditional Write **

* LS Write, Data Type Dependent. These cases now examine CSR<6>.

If CSR<6> = 0, and CM = 0 Write LS<31:0> (Data Type = LONG)
If CSR<6> = 0, and CM = 1 Write LS<15:0> (Compatibility Mode. DT = WORD)

If CSR<6> = 1, the Write is conditional on the SIZE register.

If CM = 0 SIZE<1:0> = 00 Write LS<7:0> (BYTE)
= 01 Write LS<15:0> (WORD)
= 1X Write LS<31:0> (LONG)

If CM = 1 Write LS<15:0> (WORD)

** If CSR<8> = 0, No Write
If CSR<8> = 1, Write LS<31:0> (Writes incremented PC)

:3691 .page

:3692

:3693

:3694

:3695

:3696

:3697

:3698

:3699

:3700

:3701

:3702

:3703

:3704

:3705

:3706

:3707

:3708

:3709

:3710

:3711

:3712

:3713

:3714

:3715

:3716

:3717

:3718

:3719

:3720

:3721

:3722

:3723

:3724

:3725

:3726

:3727

:3728

:3729

:3730

:3731

:3732

:3733

:3734

:3735

Table 3. SKIP/JUMP Conditions

This table summarizes the Skip and Jump conditions, and the other special functions of the micro sequencer.

SCTL codes 0-F, 17-1F are Skip conditions. The codes 10-17 execute special functions. JCTL codes 0-B are Jump conditions, and C-F execute special functions.

Sctl	Function	Sctl	Function
00	Not ALU N	10	RTN+1 if No Mem Err
01	Not ALU Z	11	Loop if Not ALU Z
02	Not ALU V	12	POP Stack
03	ALU C	13	Loop if ALU C
04	Not PSL	14	Return
05	Branch False	15	No Skip (NOP)
06	RDEST	16	RTN+1
07	Not Inter. Req.	17	Loop if No Inter. and No Sync
08	ALU N	18	Console ATTN
09	ALU Z	19	Console ACK
0A	ALU V	1A	Port Interrupt Pending
0B	Not ALU C	1B	Reg. Backup Mask Flag
0C	State 0	1C	ALU N.XOR.ALU V
0D	State 1	1D	Not Memory Error
0E	Not State 0	1E	Skip
0F	Not State 1	1F	Sync

Jctl	Function	Jctl	Function
00	Not ALU N	08	ALU N
01	Not ALU Z	09	ALU Z
02	Not ALU V	0A	ALU V
03	ALU C	0B	Not ALU C
04	JUMP	0C	JSR
05	Branch False	0D	JSR if IB Valid
06	RDEST	0E	No Jump; Skip if IB Valid
07	Not Interr. Req	0F	IB Valid

:3736 .PAGE "2901 ALU Control Description"

:3737

:3738

:3739

:3740

:3741

:3742

:3743

:3744

:3745

:3746

:3747

:3748

:3749

:3750

:3751

:3752

:3753

:3754

:3755

:3756

:3757

:3758

:3759

:3760

:3761

:3762

:3763

:3764

:3765

:3766

:3767

:3768

:3769

:3770

:3771

:3772

:3773

:3774

:3775

:3776

:3777

:3778

:3779

:3780

:3781

:3782

:3783

The 2901 ALU is the heart of the CPU module. It is responsible for all CPU calculations. The control lines going to 10 through 18 control the function, input, and output of the 2901. When a failure involving the 2901 occurs, the logical place to start looking is the control lines. The following tables describe what the control signals should be for various functions. Use it whenever the control lines are to be checked. The test descriptions tell the operator what function the 2901 is supposed to be doing.

The SRC CTL on lines 10 through 12 (pins 12 through 14 respectively) control where the inputs to the internal ALU within the 2901 come from. The inputs to the internal ALU are labeled R and S. R is one operand (4 bits) and S is the other operand (also 4 bits). The R input can come from a Working Register (labeled A), the D bus (direct input), or be forced to 0. The S input can come from either set of working registers (labeled A and B), the Q register, or forced to 0. A table of these control signals follow:

			octal code	SRC operands	
12	11	10		R	S
L	L	L	0	A	Q
L	L	H	1	A	B
L	H	L	2	0	Q
L	H	H	3	0	B
H	L	L	4	0	A
H	L	H	5	D	A
H	H	L	6	D	Q
H	H	H	7	D	0

ALU SOURCE OPERAND CONTROL

The ALU CTL on lines 13 through 15 (pins 26, 28, 27 respectively) control what function the ALU is to perform on the two operands R and S. A table follows:

3784 page

15	14	13	octal code	ALU function
L	L	L	0	R PLUS S
L	L	H	1	S MINUS R
L	H	L	2	R MINUS S
L	H	H	3	R OR S
H	L	L	4	R AND S
H	L	H	5	R NOT AND S
H	H	L	6	R XOR S
H	H	H	7	R XNOR S

ALU FUNCTION CONTROL

The DST CTL on lines 16 through 18 (pins 5, 7, 6 respectively) control what goes out onto the Y bus (either the internal ALU result or a WR directly) and what happens internal to the 2901 (shifting, writing the Q register etc). Internal shifting left or right is accomplished with this control. Below is a table explaining the various destination possibilities. The arrow refers to where the output will end up. B indicates the WR addressed by the B ADDR, F refers to the output of the internal ALU, and A refers to the output of the WR addressed by the A ADDR. Up is toward the MSB, while down is toward the LSB.

nomenclature in listing	18	17	16	octal code	RAM function shift	load	Q-reg function shift	load	Y output
LOAD Q	L	L	L	0	NONE	NONE	NONE	F->Q	F
NOP	L	L	H	1	NONE	NONE	NONE	NONE	F
WRITE.B.A	L	H	L	2	NONE	F->B	NONE	NONE	A
WRITE.B.F	L	H	H	3	NONE	F->B	NONE	NONE	F
RSHF.RAM.Q	H	L	L	4	DOWN	F/2->B	DOWN	Q/2->Q	F
RSHF.RAM	H	L	H	5	DOWN	F/2->B	NONE	NONE	F
LSHF.RAM.Q	H	H	L	6	UP	2F->B	UP	2Q->Q	F
LSHF.RAM	H	H	H	7	UP	2F->B	NONE	NONE	F

ALU DESTINATION CONTROL

3837

```

:3838 .page
:3839 LIST.TOC 'DIAGNOSTIC MACROS'
:3840
:3841 LS.DATA.WORD/=<15:0>
:3842 UPPER.BYTE/=<23:16>
:3843
:3844 WORD[] 'UPPER.BYTE/0,LS.DATA.WORD/@1'
:3845 COUNT.LS[] 'UPPER.BYTE/0,LS.DATA.WORD/@1'
:3846 COUNT.MM[] 'UPPER.BYTE/1,LS.DATA.WORD/@1'
:3847 START.ADR[] 'UPPER.BYTE/0,LS.DATA.WORD/@1'
:3848
:3849 ASCII.LIT/=<15:0>
:3850
:3851 CA=4341 ;ASCII VALUE FOR FILE NAMES
:3852 CB=4342 :
:3853 CC=4343 :
:3854 CD=4344 :
:3855 CE=4345 : V
:3856 CF=4346
:3857 CG=4347
:3858 CH=4348
:3859
:3860 ASCII [] 'UPPER.BYTE/0,ASCII.LIT/@1'
:3861
:3862 .REGION/0,6F

```

;3863 .PAGE "TEST POINTERS"

This portion contains the pointers to all tests in this section. The WCS address of the JUMP instruction corresponds to the test number. E.g. WCS 5 contains a JUMP to test 5. All unused test numbers will contain a JUMP to an error routine. This portion is 63. locations long, allowing for up to 63. tests per section, from WCS address 1 to WCS address 3F.

```

:TEST POINTERS BEGIN AT WCS ADDRESS 1

```

U 0001,	8871.94	:3874	JMP [T.1]
U 0002,	8952.04	:3875	JMP [T.2]
U 0003,	0956.F4	:3876	JMP [T.3]
U 0004,	895A.24	:3877	JMP [T.4]
U 0005,	8960.14	:3878	JMP [T.5]
U 0006,	8965.24	:3879	JMP [T.6]
U 0007,	8972.14	:3880	JMP [T.7]
U 0008,	8978.74	:3881	JMP [T.8]
U 0009,	8983.54	:3882	JMP [T.9]
U 000A,	0985.24	:3883	JMP [T.A]
U 000B,	898A.64	:3884	JMP [T.B]
U 000C,	098C.B4	:3885	JMP [T.C]
U 000D,	8993.B4	:3886	JMP [T.D]
U 000E,	8996.74	:3887	JMP [T.E]
U 000F,	0997.E4	:3888	JMP [T.F]
U 0010,	886B.E4	:3889	JMP [ERR.NO.TEST]
		:3890	
U 0011,	886B.E4	:3891	JMP [ERR.NO.TEST]
		:3892	
U 0012,	886B.E4	:3893	JMP [ERR.NO.TEST]
		:3894	
U 0013,	886B.E4	:3895	JMP [ERR.NO.TEST]
		:3896	
U 0014,	886B.E4	:3897	JMP [ERR.NO.TEST]
		:3898	
U 0015,	8869.E4	:3899	JMP [ERR.NO.TEST]
		:3900	
U 0016,	886B.E4	:3901	JMP [ERR.NO.TEST]
		:3902	
U 0017,	886B.E4	:3903	JMP [ERR.NO.TEST]
		:3904	
U 0018,	886B.E4	:3905	JMP [ERR.NO.TEST]
		:3906	
U 0019,	886B.E4	:3907	JMP [ERR.NO.TEST]
		:3908	
U 001A,	886B.E4	:3909	JMP [ERR.NO.TEST]
		:3910	
U 001B,	886B.E4	:3911	JMP [ERR.NO.TEST]
		:3912	
U 001C,	886B.E4	:3913	JMP [ERR.NO.TEST]
		:3914	
U 001D,	886B.E4	:3915	JMP [ERR.NO.TEST]
		:3916	
U 001E,	886B.E4	:3917	JMP [ERR.NO.TEST]

[illegible][illegible]

U 001F, 886B,E4	:3918	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3919		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0020, 886B,E4	:3920	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3921		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0021, 886B,E4	:3922	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3923		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0022, 886B,E4	:3924	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3925		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0023, 886B,E4	:3926	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3927		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0024, 886B,E4	:3928	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3929		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0025, 886B,E4	:3930	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3931		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0026, 886B,E4	:3932	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3933		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0027, 886B,E4	:3934	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3935		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0028, 886B,E4	:3936	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3937		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0029, 886B,E4	:3938	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3939		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 002A, 886B,E4	:3940	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3941		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 002B, 886B,E4	:3942	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3943		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 002C, 886B,E4	:3944	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3945		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 002D, 886B,E4	:3946	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3947		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 002E, 886B,E4	:3948	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3949		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 002F, 886B,E4	:3950	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3951		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0030, 886B,E4	:3952	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3953		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0031, 886B,E4	:3954	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3955		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0032, 886B,E4	:3956	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3957		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0033, 886B,E4	:3958	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3959		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0034, 886B,E4	:3960	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3961		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0035, 886B,E4	:3962	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3963		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0036, 886B,E4	:3964	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3965		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0037, 886B,E4	:3966	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3967		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0038, 886B,E4	:3968	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3969		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
U 0039, 886B,E4	:3970	JMP [ERR.NO.TEST]	: IN THIS OVERLAY
	:3971		: GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:3972		: IN THIS OVERLAY

ZZ-ENKCB-1.0	:	ENKCB.MIC							
:	:	ENKCB.MCR	MICRO2 1L(03)	TEST POINTERS	3-MAR-82	10:02:46	F 8	Fiche 1 Frame F8	Sequence 96
:	:	ENKCB.MIC	TEST POINTERS		3-MAR-82	10:02:46	ENKCB Micro-diagnostic for DAP module	Rev 01.00	Page 90

U 003A, 886B.E4	:	3973	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	3974		:	IN THIS OVERLAY
U 003B, 886B.E4	:	3975	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	3976		:	IN THIS OVERLAY
U 003C, 886B.E4	:	3977	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	3978		:	IN THIS OVERLAY
L 003D, 886B.E4	:	3979	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	3980		:	IN THIS OVERLAY
U 003E, 886B.E4	:	3981	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	3982		:	IN THIS OVERLAY
U 003F, 886B.E4	:	3983	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	3984		:	IN THIS OVERLAY
U 0040, 886B.E4	:	3985	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	3986		:	IN THIS OVERLAY
U 0041, 886B.E4	:	3987	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	3988		:	IN THIS OVERLAY
U 0042, 886B.E4	:	3989	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	3990		:	IN THIS OVERLAY
U 0043, 886B.E4	:	3991	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	3992		:	IN THIS OVERLAY
U 0044, 886B.E4	:	3993	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	3994		:	IN THIS OVERLAY
U 0045, 886B.E4	:	3995	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	3996		:	IN THIS OVERLAY
U 0046, 886B.E4	:	3997	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	3998		:	IN THIS OVERLAY
U 0047, 886B.E4	:	3999	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	4000		:	IN THIS OVERLAY
U 0048, 886B.E4	:	4001	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	4002		:	IN THIS OVERLAY
U 0049, 886B.E4	:	4003	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	4004		:	IN THIS OVERLAY
U 004A, 886B.E4	:	4005	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	4006		:	IN THIS OVERLAY
U 004B, 886B.E4	:	4007	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	4008		:	IN THIS OVERLAY
U 004C, 886B.E4	:	4009	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	4010		:	IN THIS OVERLAY
U 004D, 886B.E4	:	4011	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	4012		:	IN THIS OVERLAY
U 004E, 886B.E4	:	4013	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	4014		:	IN THIS OVERLAY
U 004F, 886B.E4	:	4015	JMP [ERR.NO.TEST]	:	GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:	4016		:	IN THIS OVERLAY

```

:4017 .PAGE "DIAGNOSTIC POINTERS"
:4018
:4019 This section contains all the pointers needed for this diagnostic
:4020 overlay. The first pointer (at WCS location 0) points to a data
:4021 transfer routine. It is the first location executed after a new
:4022 WCS load. The instruction here is a JUMP to TRANSFER.POINT which may
:4023 contain instructions to transfer data. If no data is to be trans-
:4024 ferred, or at the completion of the data transfers, the CPU will
:4025 issue CPU ATTN with all bits of the control and status wr ' clear.
:4026
:4027 The next 80. locations (from WCS location 1 to WCS location 4F)
:4028 contain pointers to each test in this overlay. The pointers are
:4029 JUMP instructions to the test number corresponding to the location
:4030 number. E.g. location 5 will contain a JUMP to test 5. All unused
:4031 test numbers will contain a JUMP to an error routine. This portion
:4032 appears after the definitions in this listing.
:4033
:4034 Finally the next 32. locations (from WCS location 50 to 6F) contain
:4035 pointers (Jump instructions) to WCS subroutines which are to be used by
:4036 the console diagnostic monitor.
:4037
U 0000, 086C,24 :4038 0: JUMP [TRANSFER.POINT] ;GO TO ROUTINE THAT LOADS LS 7 WITH
:4039 ; POINTER TO DATA SECTION AND ISSUES
:4040 ; CPU ATTN WITH TRANSFER BIT SET.
:4041
:4042 50: ;START AT WCS LOCATION 50 FOR SUB-
:4043 ; ROUTINE POINTERS
:4044
:4045 SUBROUTINE POINTERS
:4046
U 0050, 0860,84 :4047 JUMP [DEPOSIT.CSR] ;GO TO WCS SUBROUTINE WHICH WRITES THE
:4048 ; MCT CSR FOR DEPOSIT CSR COMMAND
U 0051, 0860,D4 :4049 JUMP [EXAMINE.CSR] ;GO TO WCS SUBROUTINE WHICH READS THE
:4050 ; MCT CSR FOR EXAMINE CSR COMMAND
U 0052, 0861,F4 :4051 JUMP [DEPOSIT.TB] ;GO TO WCS SUBROUTINE WHICH WRITES THE
:4052 ; MCT TRANSLATION BUFFER FOR DEPOSIT
:4053 ; TB COMMAND
U 0053, 8863,A4 :4054 JUMP [EXAMINE.TB] ;GO TO WCS SUBROUTINE WHICH READS THE
:4055 ; MCT TRANSLATION BUFFER FOR EXAMINE
:4056 ; TB COMMAND
U 0054, 8865,C4 :4057 JUMP [DEPOSIT.MM] ;GO TO WCS SUBROUTINE WHICH WRITES MAIN
:4058 ; MEMORY FOR DEPOSIT MM COMMAND. ALSO
:4059 ; USED TRANSFERING DATA FROM WCS TO
:4060 ; MAIN MEMORY.
U 0055, 0866,24 :4061 JUMP [EXAMINE.MM] ;GO TO WCS SUBROUTINE WHICH READS MAIN
:4062 ; MEMORY FOR EXAMINE MM COMMAND.
U 0056, 0868,94 :4063 JUMP [SAVE.WR] ;GO TO WCS SUBROUTINE WHICH STORES THE
:4064 ; CONTENTS OF WR0-WR3 IN LS 0-3
U 0057, 8868,E4 :4065 JUMP [RESTORE.WR] ;GO TO WCS SUBROUTINE WHICH RESTORES THE
:4066 ; CONTENTS OF WR0-WR3 FROM LS 0-3
U 0058, 8864,24 :4067 JUMP [DEPOSIT.UBS] ;GO TO WCS SUBROUTINE WHICH WRITES THE
:4068 ; UNIBUS MAP ON THE MEMORY CONTROLLER
U 0059, 0865,44 :4069 JUMP [EXAMINE.UBS] ;GO TO WCS SUBROUTINE WHICH READS THE
:4070 ; UNIBUS MAP ON THE MEMORY CONTROLLER
U 005A, 0866,84 :4071 JUMP [EXAMINE.ICSR] ;GO TO WCS SUBROUTINE WHICH READS THE

```

U 005B, 0866,B4	:4072	JMP [EXAMINE.IDAR]	: ILC CSR
	:4073		: GO TO WCS SUBROUTINE WHICH READS THE
U 005C, 0866,E4	:4074	JMP [EXAMINE.DBUF]	: IDC DAR
	:4075		: GO TO WCS SUBROUTINE WHICH READS THE
U 005D, 8867,14	:4076	JMP [EXAMINE.PATT]	: IDC DBUF
	:4077		: GO TO WCS SUBROUTINE WHICH READS THE
U 005E, 8867,44	:4078	JMP [EXAMINE.POSIT]	: IDC ECC PATTERN
	:4079		: GO TO WCS SUBROUTINE WHICH READS THE
U 005F, 0867,94	:4080	JMP [DEPOSIT.IDAR]	: IDC ECC POSITION
	:4081		: GO TO WCS SUBROUTINE WHICH WRITES THE
U 0060, 0867,C4	:4082	JMP [DEPOSIT.IDAR]	: IDC CSR
	:4083		: GO TO WCS SUBROUTINE WHICH WRITES THE
U 0061, 0867,F4	:4084	JMP [DEPOSIT.DBUF]	: IDC DAR
	:4085		: GO TO WCS SUBROUTINE WHICH WRITES THE
U 0062, 8868,24	:4086	JMP [CLEAR.FIFO.ADDR]	: IDC DBUF
	:4087		: GO TO THE WCS SUBROUTINE WHICH CLEARS
U 0063, 8868,44	:4088	JMP [SELECT.FIFO.A]	: THE IDC FIFO ADDRESS
	:4089		: GO TO THE WCS SUBROUTINE WHICH SELECTS
U 0064, 0868,64	:4090	JMP [SELECT.FIFO.B]	: FIFO A
	:4091		: GO TO THE WCS SUBROUTINE WHICH SELECTS
U 0065, DB00,15	:4092	NOP	: FIFO B
	:4093	NOP	: NOT USED
U 0066, DB00,15	:4094	NOP	: NOT USED
	:4095	NOP	: NOT USED
U 0067, DB00,15	:4096	NOP	: NOT USED
	:4097	NOP	: NOT USED
U 0068, DB00,15	:4098	NOP	: NOT USED
	:4099	NOP	: NOT USED
U 006A, DB00,15	:4100	NOP	: NOT USED
	:4101	NOP	: NOT USED
U 006B, DB00,15	:4102	NOP	: NOT USED
	:4103	NOP	: NOT USED

:4104 .PAGE
 :4105 .REGION/70,5FF
 :4106 .TOC 'LS DATA SECTION'
 :4107

:4108 :+++
 :4109 This section lists the data that will be transferred to LS by the con-
 :4110 sole as part of the initialization performed in test 0. The TRANSFER
 :4111 DATA routine will point to this data area. LS contains the constants,
 :4112 control and status word, and temporary storage locations used by the
 :4113 tests and subroutines of the microdiagnostic.
 :4114 The LS is 32 bits wide. Each of these words is 16 bits, so two con-
 :4115 secutive words listed here will be loaded into each consecutive LS loc-
 :4116 ation. The first word listed will be bits 0-15 of the LS location, the
 :4117 second word listed will be bits 16-31 of the same LS location.
 :4118 The locations 78-7F in the first half of LS and F8-FF in the second
 :4119 half of LS are not actual LS locations. They force an access to one of
 :4120 several registers in the CPU or force an indexing feature to another LS
 :4121 location. For that reason, they are not written via the transfer rou-
 :4122 tine.
 :4123 Note: This table is in hexadecimal format i.e. each digit represents
 :4124 four bits, so four digits represents a full 16 bit word. The micro-
 :4125 assembler requires that a hexadecimal value that starts with a letter
 :4126 (e.g. FFFF) be preceded by a 0 (0FFFF). So some table values will
 :4127 contain 5 hexadecimal digits. The fifth digit is not actually used.
 :4128 :---
 :4129
 :4130

TRANSFER.DATA.LS1:

U 0070, 0000,78	COUNT.LS[0078]	:LONGWORD COUNT (NUMBER OF LONGWORDS TO TRANS-
U 0071, 0000,00	START.ADR[0000]	:FER TO LS = 120 DECIMAL) DESTINATION IS LS
U 0072, 0000,00	WORD[0000]	:DESTINATION START ADDRESS (START AT LS 0)
U 0073, 0000,00	WORD[0000]	:LOCATION 0
U 0074, 0000,00	WORD[0000]	:LOCATION 1
U 0075, 0000,00	WORD[0000]	:LOCATION 2
U 0076, 0000,00	WORD[0000]	:LOCATION 3
U 0077, 0000,00	WORD[0000]	:LOCATION 4
U 0078, 0000,00	WORD[0000]	:LOCATION 5
U 0079, 0000,00	WORD[0000]	:LOCATION 6
U 007A, 0000,00	WORD[0000]	:LOCATION 7
U 007B, 0000,00	WORD[0000]	
U 007C, 0000,00	WORD[0000]	
U 007D, 0000,00	WORD[0000]	
U 007E, 0000,00	WORD[0000]	
U 007F, 0000,00	WORD[0000]	
U 0080, 0000,00	WORD[0000]	

ZZ-ENKCB-1.0	:	ENKCB.MIC							
:	:	ENKCB.MCR	MICRO2 1L(03)	LS DATA SECTION	3-MAR-82	10:02:46	J 8	Fiche 1 Frame J8	Sequence 100
:	:	ENKCB.MIC	LS DATA SECTION					ENKCB Micro-diagnostic for DAP module	Rev 01.00 Page 94

U 0081, 0000,00	:	4159	WORD[0000]	
	:	4160		
U 0082, 0000,00	:	4161	WORD[0000]	;LOCATION 8
U 0083, 0000,00	:	4162	WORD[0000]	
	:	4163		
U 0084, 0000,00	:	4164	WORD[0000]	;LOCATION 9
U 0085, 0000,00	:	4165	WORD[0000]	
	:	4166		
U 0086, 0000,00	:	4167	WORD[0000]	;LOCATION 0A
U 0087, 0000,00	:	4168	WORD[0000]	
	:	4169		
U 0088, 0000,00	:	4170	WORD[0000]	;LOCATION 0B
U 0089, 0000,00	:	4171	WORD[0000]	
	:	4172		
U 008A, 0000,00	:	4173	WORD[0000]	;LOCATION 0C
U 008B, 0000,00	:	4174	WORD[0000]	
	:	4175		
U 008C, 0000,00	:	4176	WORD[0000]	;LOCATION 0D
U 008D, 0000,00	:	4177	WORD[0000]	
	:	4178		
U 008E, 0000,00	:	4179	WORD[0000]	;LOCATION 0E
U 008F, 0000,00	:	4180	WORD[0000]	
	:	4181		
U 0090, 0000,00	:	4182	WORD[0000]	;LOCATION 0F
U 0091, 0000,00	:	4183	WORD[0000]	
	:	4184		
U 0092, 0000,00	:	4185	WORD[0000]	;LOCATION 10
U 0093, 0000,00	:	4186	WORD[0000]	
	:	4187		
U 0094, 0000,00	:	4188	WORD[0000]	;LOCATION 11
U 0095, 0000,00	:	4189	WORD[0000]	
	:	4190		
U 0096, 0000,00	:	4191	WORD[0000]	;LOCATION 12
U 0097, 0000,00	:	4192	WORD[0000]	
	:	4193		
U 0098, 0000,FF	:	4194	WORD[00FF]	;LOCATION 13
U 0099, 0000,00	:	4195	WORD[0000]	
	:	4196		
U 009A, 00FF,FF	:	4197	WORD[0FFF]	;LOCATION 14
U 009B, 0000,00	:	4198	WORD[0000]	
	:	4199		
U 009C, 0000,00	:	4200	WORD[0000]	;LOCATION 15
U 009D, 00FF,00	:	4201	WORD[0FF00]	
	:	4202		
U 009E, 00FF,00	:	4203	WORD[0FF00]	;LOCATION 16
U 009F, 00FF,FF	:	4204	WORD[0FFFF]	
	:	4205		
U 00A0, 003F,FF	:	4206	WORD[3FFF]	;LOCATION 17
U 00A1, 0001,00	:	4207	WORD[0100]	
	:	4208		
U 00A2, 003F,FF	:	4209	WORD[3FFF]	;LOCATION 18
U 00A3, 00FF,FE	:	4210	WORD[0FFFE]	
	:	4211		
U 00A4, 00FF,FF	:	4212	WORD[0FFFF]	;LOCATION 19
U 00A5, 00FE,7F	:	4213	WORD[0FE7F]	

U 00A6, 0000,00	:4214	WORD[0000]	:LOCATION 1A
U 00A7, 007F,F8	:4215	WORD[7FF8]	
	:4216		
U 00A8, 0080,00	:4217	WORD[8000]	:LOCATION 1B
U 00A9, 007F,FF	:4218	WORD[7FFF]	
	:4219		
U 00AA, 0000,00	:4220	WORD[0000]	:LOCATION 1C
U 00AB, 0000,00	:4221	WORD[0000]	
	:4222		
U 00AC, 0000,00	:4223	WORD[0000]	:LOCATION 1D
U 00AD, 0000,00	:4224	WORD[0000]	
	:4225		
U 00AE, 0005,00	:4226	WORD[0500]	:LOCATION 1E
U 00AF, 0000,80	:4227	WORD[0080]	
	:4228		
U 00B0, 0005,00	:4229	WORD[0500]	:LOCATION 1F
U 00B1, 0000,00	:4230	WORD[0000]	
	:4231		
U 00B2, 0000,01	:4232	WORD[0001]	:LOCATION 20
U 00B3, 0000,00	:4233	WORD[0000]	
	:4234		
U 00B4, 0000,02	:4235	WORD[0002]	:LOCATION 21
U 00B5, 0000,00	:4236	WORD[0000]	
	:4237		
U 00B6, 0000,04	:4238	WORD[0004]	:LOCATION 22
U 00B7, 0000,00	:4239	WORD[0000]	
	:4240		
U 00B8, 0000,08	:4241	WORD[0008]	:LOCATION 23
U 00B9, 0000,00	:4242	WORD[0000]	
	:4243		
U 00BA, 0000,10	:4244	WORD[0010]	:LOCATION 24
U 00BB, 0000,00	:4245	WORD[0000]	
	:4246		
U 00BC, 0000,20	:4247	WORD[0020]	:LOCATION 25
U 00BD, 0000,00	:4248	WORD[0000]	
	:4249		
U 00BE, 0000,40	:4250	WORD[0040]	:LOCATION 26
U 00BF, 0000,00	:4251	WORD[0000]	
	:4252		
U 00C0, 0000,80	:4253	WORD[0080]	:LOCATION 27
U 00C1, 0000,00	:4254	WORD[0000]	
	:4255		
U 00C2, 0001,00	:4256	WORD[0100]	:LOCATION 28
U 00C3, 0000,00	:4257	WORD[0000]	
	:4258		
U 00C4, 0002,00	:4259	WORD[0200]	:LOCATION 29
U 00C5, 0000,00	:4260	WORD[0000]	
	:4261		
U 00C6, 0004,00	:4262	WORD[0400]	:LOCATION 2A
U 00C7, 0000,00	:4263	WORD[0000]	
	:4264		
U 00C8, 0008,00	:4265	WORD[0800]	:LOCATION 2B
U 00C9, 0000,00	:4266	WORD[0000]	
	:4267		
	:4268		

U 00CA, 0010,00	:4269	WORD[1000]	:LOCATION 2C
U 00CB, 0000,00	:4270	WORD[0000]	
	:4271		
U 00CC, 0020,00	:4272	WORD[2000]	:LOCATION 2D
U 00CD, 0000,00	:4273	WORD[0000]	
	:4274		
U 00CE, 0040,00	:4275	WORD[4000]	:LOCATION 2E
U 00CF, 0000,00	:4276	WORD[0000]	
	:4277		
U 00D0, 0080,00	:4278	WORD[8000]	:LOCATION 2F
U 00D1, 0000,00	:4279	WORD[0000]	
	:4280		
U 00D2, 0000,00	:4281	WORD[0000]	:LOCATION 30
U 00D3, 0000,01	:4282	WORD[0001]	
	:4283		
U 00D4, 0000,00	:4284	WORD[0000]	:LOCATION 31
U 00D5, 0000,02	:4285	WORD[0002]	
	:4286		
U 00D6, 0000,00	:4287	WORD[0000]	:LOCATION 32
U 00D7, 0000,04	:4288	WORD[0004]	
	:4289		
U 00D8, 0000,00	:4290	WORD[0000]	:LOCATION 33
U 00D9, 0000,08	:4291	WORD[0008]	
	:4292		
U 00DA, 0000,00	:4293	WORD[0000]	:LOCATION 34
U 00DB, 0000,10	:4294	WORD[0010]	
	:4295		
U 00DC, 0000,00	:4296	WORD[0000]	:LOCATION 35
U 00DD, 0000,20	:4297	WORD[0020]	
	:4298		
U 00DE, 0000,00	:4299	WORD[0000]	:LOCATION 36
U 00DF, 0000,40	:4300	WORD[0040]	
	:4301		
U 00E0, 0000,00	:4302	WORD[0000]	:LOCATION 37
U 00E1, 0000,80	:4303	WORD[0080]	
	:4304		
U 00E2, 0000,00	:4305	WORD[0000]	:LOCATION 38
U 00E3, 0001,00	:4306	WORD[0100]	
	:4307		
U 00E4, 0000,00	:4308	WORD[0000]	:LOCATION 39
U 00E5, 0002,00	:4309	WORD[0200]	
	:4310		
U 00E6, 0000,00	:4311	WORD[0000]	:LOCATION 3A
U 00E7, 0004,00	:4312	WORD[0400]	
	:4313		
U 00E8, 0000,00	:4314	WORD[0000]	:LOCATION 3B
U 00E9, 0008,00	:4315	WORD[0800]	
	:4316		
U 00EA, 0000,00	:4317	WORD[0000]	:LOCATION 3C
U 00EB, 0010,00	:4318	WORD[1000]	
	:4319		
U 00EC, 0000,00	:4320	WORD[0000]	:LOCATION 3D
U 00ED, 0020,00	:4321	WORD[2000]	
	:4322		
U 00EE, 0000,00	:4323	WORD[0000]	:LOCATION 3E

U 00EF, 0040,00	:4324	WORD[4000]	
U 00F0, 0000,00	:4325	WORD[0000]	;LOCATION 3F
U 00F1, 0080,00	:4326	WORD[8000]	
U 00F2, 0000,00	:4327	WORD[0000]	;LOCATION 40
U 00F3, 0000,00	:4328	WORD[0000]	
U 00F4, 0000,00	:4329	WORD[0000]	;LOCATION 41
U 00F5, 0000,00	:4330	WORD[0000]	
U 00F6, 0000,00	:4331	WORD[0000]	;LOCATION 42
U 00F7, 0000,00	:4332	WORD[0000]	
U 00F8, 0000,00	:4333	WORD[0000]	;LOCATION 43
U 00F9, 0000,00	:4334	WORD[0000]	
U 00FA, 0000,00	:4335	WORD[0000]	;LOCATION 44
U 00FB, 0000,00	:4336	WORD[0000]	
U 00FC, 0000,00	:4337	WORD[0000]	;LOCATION 45
U 00FD, 0000,00	:4338	WORD[0000]	
U 00FE, 0000,00	:4339	WORD[0000]	;LOCATION 46
U 00FF, 0000,00	:4340	WORD[0000]	
U 0100, 0000,00	:4341	WORD[0000]	;LOCATION 47
U 0101, 0000,00	:4342	WORD[0000]	
U 0102, 0000,00	:4343	WORD[0000]	;LOCATION 48
U 0103, 0000,00	:4344	WORD[0000]	
U 0104, 0000,00	:4345	WORD[0000]	;LOCATION 49
U 0105, 0000,00	:4346	WORD[0000]	
U 0106, 0000,00	:4347	WORD[0000]	;LOCATION 4A
U 0107, 0000,00	:4348	WORD[0000]	
U 0108, 0055,55	:4349	WORD[5555]	;LOCATION 4B
U 0109, 00AA,AA	:4350	WORD[0AAAA]	
U 010A, 00AA,AA	:4351	WORD[0AAAA]	;LOCATION 4C
U 010B, 00AA,AA	:4352	WORD[0AAAA]	
U 010C, 0055,55	:4353	WORD[5555]	;LOCATION 4D
U 010D, 0055,55	:4354	WORD[5555]	
U 010E, 0000,00	:4355	WORD[0000]	;LOCATION 4E
U 010F, 0000,00	:4356	WORD[0000]	
U 0110, 00FF,FF	:4357	WORD[0FFFF]	;LOCATION 4F
U 0111, 00FF,FF	:4358	WORD[0FFFF]	
U 0112, 0000,00	:4359	WORD[0000]	;LOCATION 50
U 0113, 0000,00	:4360	WORD[0000]	

U 0114, 0000,00	:4379		
U 0115, 0000,00	:4380	WORD[0000]	;LOCATION 51
	:4381	WORD[0000]	
	:4382		
U 0116, 0000,00	:4383	WORD[0000]	;LOCATION 52
U 0117, 0000,00	:4384	WORD[0000]	
	:4385		
U 0118, 0000,00	:4386	WORD[0000]	;LOCATION 53
U 0119, 0000,00	:4387	WORD[0000]	
	:4388		
U 011A, 0000,00	:4389	WORD[0000]	;LOCATION 54
U 011B, 0000,00	:4390	WORD[0000]	
	:4391		
U 011C, 0000,00	:4392	WORD[0000]	;LOCATION 55
U 011D, 0000,00	:4393	WORD[0000]	
	:4394		
U 011E, 0000,00	:4395	WORD[0000]	;LOCATION 56
U 011F, 0000,00	:4396	WORD[0000]	
	:4397		
U 0120, 0000,00	:4398	WORD[0000]	;LOCATION 57
U 0121, 0000,00	:4399	WORD[0000]	
	:4400		
U 0122, 0000,00	:4401	WORD[0000]	;LOCATION 58
U 0123, 0000,00	:4402	WORD[0000]	
	:4403		
U 0124, 0000,00	:4404	WORD[0000]	;LOCATION 59
U 0125, 0000,00	:4405	WORD[0000]	
	:4406		
U 0126, 0000,00	:4407	WORD[0000]	;LOCATION 5A
U 0127, 0000,00	:4408	WORD[0000]	
	:4409		
U 0128, 0000,00	:4410	WORD[0000]	;LOCATION 5B
U 0129, 0000,00	:4411	WORD[0000]	
	:4412		
U 012A, 0000,00	:4413	WORD[0000]	;LOCATION 5C
U 012B, 0000,00	:4414	WORD[0000]	
	:4415		
U 012C, 0000,00	:4416	WORD[0000]	;LOCATION 5D
U 012D, 0000,00	:4417	WORD[0000]	
	:4418		
U 012E, 0000,00	:4419	WORD[0000]	;LOCATION 5E
U 012F, 0000,00	:4420	WORD[0000]	
	:4421		
U 0130, 0000,00	:4422	WORD[0000]	;LOCATION 5F
U 0131, 0000,00	:4423	WORD[0000]	
	:4424		
U 0132, 0000,03	:4425	WORD[0003]	;LOCATION 60
U 0133, 0000,00	:4426	WORD[0000]	
	:4427		
U 0134, 0000,12	:4428	WORD[0012]	;LOCATION 61
U 0135, 0000,00	:4429	WORD[0000]	
	:4430		
U 0136, 0033,33	:4431	WORD[3333]	;LOCATION 62
U 0137, 0033,33	:4432	WORD[3333]	
	:4433		

ZZ-ENKCB-1.0	:	ENKCB.MIC							
:		ENKCB.MIC							
:		ENKCB.MIC							
			MICRO2 1L(03)	LS DATA SECTION	3-MAR-82	10:02:46	ENKCB Micro-diagnostic for DAP module	Rev 01.00	Page 99
U 0138,	000F,0F	:4434	WORD[0F0F]						
U 0139,	000F,0F	:4435	WORD[0F0F]						
		:4436							
U 013A,	0000,FF	:4437	WORD[00FF]						
U 013B,	0000,FF	:4438	WORD[00FF]						
		:4439							
U 013C,	0001,62	:4440	WORD[0162]						
U 013D,	0000,00	:4441	WORD[0000]						
		:4442							
U 013E,	0000,00	:4443	WORD[0000]						
U 013F,	0000,00	:4444	WORD[0000]						
		:4445							
U 0140,	0000,00	:4446	WORD[0000]						
U 0141,	0000,00	:4447	WORD[0000]						
		:4448							
U 0142,	0000,00	:4449	WORD[0000]						
U 0143,	0000,00	:4450	WORD[0000]						
		:4451							
U 0144,	0000,00	:4452	WORD[0000]						
U 0145,	0000,00	:4453	WORD[0000]						
		:4454							
U 0146,	0000,00	:4455	WORD[0000]						
U 0147,	0000,00	:4456	WORD[0000]						
		:4457							
U 0148,	0000,00	:4458	WORD[0000]						
U 0149,	0000,00	:4459	WORD[0000]						
		:4460							
U 014A,	0000,00	:4461	WORD[0000]						
U 014B,	0000,00	:4462	WORD[0000]						
		:4463							
U 014C,	0000,00	:4464	WORD[0000]						
U 014D,	0000,00	:4465	WORD[0000]						
		:4466							
U 014E,	0000,00	:4467	WORD[0000]						
U 014F,	0000,00	:4468	WORD[0000]						
		:4469							
U 0150,	0000,00	:4470	WORD[0000]						
U 0151,	0000,00	:4471	WORD[0000]						
		:4472							
U 0152,	0000,00	:4473	WORD[0000]						
U 0153,	0000,00	:4474	WORD[0000]						
		:4475							
U 0154,	0000,00	:4476	WORD[0000]						
U 0155,	0000,00	:4477	WORD[0000]						
		:4478							
U 0156,	0000,00	:4479	WORD[0000]						
U 0157,	0000,00	:4480	WORD[0000]						
		:4481							
U 0158,	0000,00	:4482	WORD[0000]						
U 0159,	0000,00	:4483	WORD[0000]						
		:4484							
U 015A,	0000,00	:4485	WORD[0000]						
U 015B,	0000,00	:4486	WORD[0000]						
		:4487							
U 015C,	0000,00	:4488	WORD[0000]						

U 017C, 0000,00	:4544	WORD[0000]	;LOCATION 8C
U 017D, 0000,00	:4545	WORD[0000]	
	:4546		
U 017E, 0000,00	:4547	WORD[0000]	;LOCATION 8D
U 017F, 0000,00	:4548	WORD[0000]	
	:4549		
U 0180, 0000,00	:4550	WORD[0000]	;LOCATION 8E
U 0181, 0000,00	:4551	WORD[0000]	
	:4552		
U 0182, 0000,00	:4553	WORD[0000]	;LOCATION 8F
U 0183, 0000,00	:4554	WORD[0000]	
	:4555		
U 0184, 0000,00	:4556	WORD[0000]	;LOCATION 90
U 0185, 0000,00	:4557	WORD[0000]	
	:4558		
U 0186, 0055,55	:4559	WORD[5555]	;LOCATION 91
U 0187, 00AA,AA	:4560	WORD[0AAAA]	
	:4561		
U 0188, 0000,00	:4562	WORD[0000]	;LOCATION 92
U 0189, 0000,1F	:4563	WORD[001F]	
	:4564		
U 018A, 0000,00	:4565	WORD[0000]	;LOCATION 93
U 018B, 0000,00	:4566	WORD[0000]	
	:4567		
U 018C, 0000,00	:4568	WORD[0000]	;LOCATION 94
U 018D, 0000,00	:4569	WORD[0000]	
	:4570		
U 018E, 00FF,FC	:4571	WORD[0FFFFC]	;LOCATION 95
U 018F, 00FF,FF	:4572	WORD[0FFFFF]	
	:4573		
U 0190, 0000,0F	:4574	WORD[000F]	;LOCATION 96
U 0191, 0000,00	:4575	WORD[0000]	
	:4576		
U 0192, 0000,F0	:4577	WORD[00F0]	;LOCATION 97
U 0193, 0000,00	:4578	WORD[0000]	
	:4579		
U 0194, 000F,00	:4580	WORD[0F00]	;LOCATION 98
U 0195, 0000,00	:4581	WORD[0000]	
	:4582		
U 0196, 00F0,00	:4583	WORD[0F000]	;LOCATION 99
U 0197, 0000,00	:4584	WORD[0000]	
	:4585		
U 0198, 0000,00	:4586	WORD[0000]	;LOCATION 9A
U 0199, 0000,03	:4587	WORD[0003]	
	:4588		
U 019A, 0080,00	:4589	WORD[8000]	;LOCATION 9B
U 019B, 007F,FF	:4590	WORD[7FFFF]	
	:4591		
U 019C, 0000,00	:4592	WORD[0000]	;LOCATION 9C
U 019D, 0000,54	:4593	WORD[0054]	
	:4594		
U 019E, 0000,00	:4595	WORD[0000]	;LOCATION 9D
U 019F, 0000,F0	:4596	WORD[00F0]	
	:4597		
U 01A0, 0000,00	:4598	WORD[0000]	;LOCATION 9E

U 01A1, 0000,FC	:4599	WORD[00FC]	
	:4600		
U 01A2, 003F,FF	:4601	WORD[3FFF]	:LOCATION 9F
U 01A3, 003F,00	:4602	WORD[3F00]	
	:4603		
U 01A4, 0002,54	:4604	WORD[0254]	:LOCATION 0A0
U 01A5, 0000,00	:4605	WORD[0000]	
	:4606		
U 01A6, 0000,00	:4607	WORD[0000]	:LOCATION 0A1
U 01A7, 0000,00	:4608	WORD[0000]	
	:4609		
U 01A8, 0000,00	:4610	WORD[0000]	:LOCATION 0A2
U 01A9, 0000,00	:4611	WORD[0000]	
	:4612		
U 01AA, 0000,00	:4613	WORD[0000]	:LOCATION 0A3
U 01AB, 0000,00	:4614	WORD[0000]	
	:4615		
U 01AC, 0000,00	:4616	WORD[0000]	:LOCATION 0A4
U 01AD, 0000,00	:4617	WORD[0000]	
	:4618		
U 01AE, 0000,00	:4619	WORD[0000]	:LOCATION 0A5
U 01AF, 0000,00	:4620	WORD[0000]	
	:4621		
U 01B0, 0000,00	:4622	WORD[0000]	:LOCATION 0A6
U 01B1, 0000,00	:4623	WORD[0000]	
	:4624		
U 01B2, 0000,00	:4625	WORD[0000]	:LOCATION 0A7
U 01B3, 0000,00	:4626	WORD[0000]	
	:4627		
U 01B4, 0000,00	:4628	WORD[0000]	:LOCATION 0A8
U 01B5, 0000,00	:4629	WORD[0000]	
	:4630		
U 01B6, 0000,00	:4631	WORD[0000]	:LOCATION 0A9
U 01B7, 0000,00	:4632	WORD[0000]	
	:4633		
U 01B8, 0000,00	:4634	WORD[0000]	:LOCATION 0AA
U 01B9, 0000,00	:4635	WORD[0000]	
	:4636		
U 01BA, 0000,00	:4637	WORD[0000]	:LOCATION 0AB
U 01BB, 0000,00	:4638	WORD[0000]	
	:4639		
U 01BC, 0000,00	:4640	WORD[0000]	:LOCATION 0AC
U 01BD, 0000,00	:4641	WORD[0000]	
	:4642		
U 01BE, 0000,00	:4643	WORD[0000]	:LOCATION 0AD
U 01BF, 0000,00	:4644	WORD[0000]	
	:4645		
U 01C0, 0000,00	:4646	WORD[0000]	:LOCATION 0AE
U 01C1, 0000,00	:4647	WORD[0000]	
	:4648		
U 01C2, 0000,00	:4649	WORD[0000]	:LOCATION 0AF
U 01C3, 0000,00	:4650	WORD[0000]	
	:4651		
U 01C4, 0000,00	:4652	WORD[0000]	:LOCATION 0B0
U 01C5, 0000,00	:4653	WORD[0000]	

U 01C6, 0000.00	:4654	WORD[0000]	:LOCATION 0B1
U 01C7, 0000.00	:4655	WORD[0000]	
	:4656		
	:4657		
U 01C8, 0000.00	:4658	WORD[0000]	:LOCATION 0B2
U 01C9, 0000.00	:4659	WORD[0000]	
	:4660		
U 01CA, 0000.00	:4661	WORD[0000]	:LOCATION 0B3
U 01CB, 0000.00	:4662	WORD[0000]	
	:4663		
U 01CC, 0000.00	:4664	WORD[0000]	:LOCATION 0B4
U 01CD, 0000.00	:4665	WORD[0000]	
	:4666		
U 01CE, 0000.00	:4667	WORD[0000]	:LOCATION 0B5
U 01CF, 0000.00	:4668	WORD[0000]	
	:4669		
U 01D0, 0000.00	:4670	WORD[0000]	:LOCATION 0B6
U 01D1, 0000.00	:4671	WORD[0000]	
	:4672		
U 01D2, 0000.00	:4673	WORD[0000]	:LOCATION 0B7
U 01D3, 0000.00	:4674	WORD[0000]	
	:4675		
U 01D4, 0000.00	:4676	WORD[0000]	:LOCATION 0B8
U 01D5, 0000.00	:4677	WORD[0000]	
	:4678		
U 01D6, 0000.00	:4679	WORD[0000]	:LOCATION 0B9
U 01D7, 0000.00	:4680	WORD[0000]	
	:4681		
U 01D8, 0000.00	:4682	WORD[0000]	:LOCATION 0BA
U 01D9, 0000.00	:4683	WORD[0000]	
	:4684		
U 01DA, 0000.00	:4685	WORD[0000]	:LOCATION 0BB
U 01DB, 0000.00	:4686	WORD[0000]	
	:4687		
U 01DC, 0000.00	:4688	WORD[0000]	:LOCATION 0BC
U 01DD, 0000.00	:4689	WORD[0000]	
	:4690		
U 01DE, 0000.00	:4691	WORD[0000]	:LOCATION 0BD
U 01DF, 0000.00	:4692	WORD[0000]	
	:4693		
U 01E0, 0000.00	:4694	WORD[0000]	:LOCATION 0BE
U 01E1, 0000.00	:4695	WORD[0000]	
	:4696		
U 01E2, 0000.00	:4697	WORD[0000]	:LOCATION 0BF
U 01E3, 0000.00	:4698	WORD[0000]	
	:4699		
U 01E4, 0000.00	:4700	WORD[0000]	:LOCATION 0C0
U 01E5, 0000.00	:4701	WORD[0000]	
	:4702		
U 01E6, 0000.00	:4703	WORD[0000]	:LOCATION 0C1
U 01E7, 0000.00	:4704	WORD[0000]	
	:4705		
U 01E8, 0000.00	:4706	WORD[0000]	:LOCATION 0C2
U 01E9, 0000.00	:4707	WORD[0000]	
	:4708		

U 01EA, 0000.00	:4709	WORD[0000]	:LOCATION 0C3
U 01EB, 0000.00	:4710	WORD[0000]	
	:4711		
U 01EC, 0000.00	:4712	WORD[0000]	:LOCATION 0C4
U 01ED, 0000.00	:4713	WORD[0000]	
	:4714		
U 01EE, 0000.00	:4715	WORD[0000]	:LOCATION 0C5
U 01EF, 0000.00	:4716	WORD[0000]	
	:4717		
U 01F0, 0000.00	:4718	WORD[0000]	:LOCATION 0C6
U 01F1, 0000.00	:4719	WORD[0000]	
	:4720		
U 01F2, 0000.00	:4721	WORD[0000]	:LOCATION 0C7
U 01F3, 0000.00	:4722	WORD[0000]	
	:4723		
U 01F4, 0000.00	:4724	WORD[0000]	:LOCATION 0C8
U 01F5, 0000.00	:4725	WORD[0000]	
	:4726		
U 01F6, 0000.00	:4727	WORD[0000]	:LOCATION 0C9
U 01F7, 0000.00	:4728	WORD[0000]	
	:4729		
U 01F8, 0000.00	:4730	WORD[0000]	:LOCATION 0CA
U 01F9, 0000.00	:4731	WORD[0000]	
	:4732		
U 01FA, 0000.00	:4733	WORD[0000]	:LOCATION 0CB
U 01FB, 0000.00	:4734	WORD[0000]	
	:4735		
U 01FC, 0000.00	:4736	WORD[0000]	:LOCATION 0CC
U 01FD, 0000.00	:4737	WORD[0000]	
	:4738		
U 01FE, 0000.00	:4739	WORD[0000]	:LOCATION 0CD
U 01FF, 0000.00	:4740	WORD[0000]	
	:4741		
U 0200, 0000.00	:4742	WORD[0000]	:LOCATION 0CE
U 0201, 0000.00	:4743	WORD[0000]	
	:4744		
U 0202, 0000.00	:4745	WORD[0000]	:LOCATION 0CF
U 0203, 0000.00	:4746	WORD[0000]	
	:4747		
U 0204, 0000.00	:4748	WORD[00,00]	:LOCATION 0D0
U 0205, 0000.00	:4749	WORD[0000]	
	:4750		
U 0206, 0000.00	:4751	WORD[0000]	:LOCATION 0D1
U 0207, 0000.00	:4752	WORD[0000]	
	:4753		
U 0208, 0000.00	:4754	WORD[0000]	:LOCATION 0D2
U 0209, 0000.00	:4755	WORD[0000]	
	:4756		
U 020A, 0000.00	:4757	WORD[0000]	:LOCATION 0D3
U 020B, 0000.00	:4758	WORD[0000]	
	:4759		
U 020C, 0000.00	:4760	WORD[0000]	:LOCATION 0D4
U 020D, 0000.00	:4761	WORD[0000]	
	:4762		
U 020E, 0000.00	:4763	WORD[0000]	:LOCATION 0D5

U 020F, 0000.00	:4764	WORD[0000]	
	:4765		
U 0210, 0000.00	:4766	WORD[0000]	:LOCATION 0D6
U 0211, 0000.00	:4767	WORD[0000]	
	:4768		
U 0212, 0000.00	:4769	WORD[0000]	:LOCATION 0D7
U 0213, 0000.00	:4770	WORD[0000]	
	:4771		
U 0214, 0000.00	:4772	WORD[0000]	:LOCATION 0D8
U 0215, 0000.00	:4773	WORD[0000]	
	:4774		
U 0216, 0000.00	:4775	WORD[0000]	:LOCATION 0D9
U 0217, 0000.00	:4776	WORD[0000]	
	:4777		
U 0218, 0000.00	:4778	WORD[0000]	:LOCATION 0DA
U 0219, 0000.00	:4779	WORD[0000]	
	:4780		
U 021A, 0000.00	:4781	WORD[0000]	:LOCATION 0DB
U 021B, 0000.00	:4782	WORD[0000]	
	:4783		
U 021C, 0000.00	:4784	WORD[0000]	:LOCATION 0DC
U 021D, 0000.00	:4785	WORD[0000]	
	:4786		
U 021E, 0000.00	:4787	WORD[0000]	:LOCATION 0DD
U 021F, 0000.00	:4788	WORD[0000]	
	:4789		
U 0220, 0000.00	:4790	WORD[0000]	:LOCATION 0DE
U 0221, 0000.00	:4791	WORD[0000]	
	:4792		
U 0222, 0000.00	:4793	WORD[0000]	:LOCATION 0DF
U 0223, 0000.00	:4794	WORD[0000]	
	:4795		
U 0224, 0000.00	:4796	WORD[0000]	:LOCATION 0E0
U 0225, 0000.00	:4797	WORD[0000]	
	:4798		
U 0226, 0000.00	:4799	WORD[0000]	:LOCATION 0E1
U 0227, 0000.00	:4800	WORD[0000]	
	:4801		
U 0228, 0000.00	:4802	WORD[0000]	:LOCATION 0E2
U 0229, 0000.00	:4803	WORD[0000]	
	:4804		
U 022A, 0000.00	:4805	WORD[0000]	:LOCATION 0E3
U 022B, 0000.00	:4806	WORD[0000]	
	:4807		
U 022C, 0000.00	:4808	WORD[0000]	:LOCATION 0E4
U 022D, 0000.00	:4809	WORD[0000]	
	:4810		
U 022E, 0000.00	:4811	WORD[0000]	:LOCATION 0E5
U 022F, 0000.00	:4812	WORD[0000]	
	:4813		
U 0230, 0000.00	:4814	WORD[0000]	:LOCATION 0E6
U 0231, 0000.00	:4815	WORD[0000]	
	:4816		
U 0232, 0000.00	:4817	WORD[0000]	:LOCATION 0E7
U 0233, 0000.00	:4818	WORD[0000]	

U 0234, 0000.00	:4819		
U 0235, 0000.00	:4820	WORD[0000]	:LOCATION 0E8
	:4821	WORD[0000]	
	:4822		
U 0236, 0000.00	:4823	WORD[0000]	:LOCATION 0E9
U 0237, 0000.00	:4824	WORD[0000]	
	:4825		
U 0238, 0000.00	:4826	WORD[0000]	:LOCATION 0EA
U 0239, 0000.00	:4827	WORD[0000]	
	:4828		
U 023A, 0000.00	:4829	WORD[0000]	:LOCATION 0EB
U 023B, 0000.00	:4830	WORD[0000]	
	:4831		
U 023C, 0000.00	:4832	WORD[0000]	:LOCATION 0EC
U 023D, 0000.00	:4833	WORD[0000]	
	:4834		
U 023E, 0000.00	:4835	WORD[0000]	:LOCATION 0ED
U 023F, 0000.00	:4836	WORD[0000]	
	:4837		
U 0240, 0000.00	:4838	WORD[0000]	:LOCATION 0EE
U 0241, 0000.00	:4839	WORD[0000]	
	:4840		
U 0242, 0000.00	:4841	WORD[0000]	:LOCATION 0EF
U 0243, 0000.00	:4842	WORD[0000]	
	:4843		
U 0244, 0000.00	:4844	WORD[0000]	:LOCATION 0F0
U 0245, 0000.00	:4845	WORD[0000]	
	:4846		
U 0246, 0000.00	:4847	WORD[0000]	:LOCATION 0F1
U 0247, 0000.00	:4848	WORD[0000]	
	:4849		
U 0248, 0000.00	:4850	WORD[0000]	:LOCATION 0F2
U 0249, 0000.00	:4851	WORD[0000]	
	:4852		
U 024A, 0000.00	:4853	WORD[0000]	:LOCATION 0F3
U 024B, 0000.00	:4854	WORD[0000]	
	:4855		
U 024C, 0000.00	:4856	WORD[0000]	:LOCATION 0F4
U 024D, 0000.00	:4857	WORD[0000]	
	:4858		
U 024E, 0000.00	:4859	WORD[0000]	:LOCATION 0F5
U 024F, 0000.00	:4860	WORD[0000]	
	:4861		
U 0250, 0000.00	:4862	WORD[0000]	:LOCATION 0F6
U 0251, 0000.00	:4863	WORD[0000]	
	:4864		
U 0252, 0000.00	:4865	WORD[0000]	:LOCATION 0F7
U 0253, 0000.00	:4866	WORD[0000]	

:IF ANY TRANSFER DATA IS ADDED, PUT BEFORE THIS
 : LINE. LIMIT OF 962 ADDITIONAL WORDS (470
 : LONGWORD TRANSFERS).

3FF:
 RETURN.3FF:

ZZ-ENKCB-1.0 : ENKCB.MIC
: ENKCB.MCR
: ENKCB.MIC
U 03FF, 5B00,14 :4874
 :4875 .REGION/600,6FF

PROGRAM SPECIFIC DATA SECTION (LS 80-F7)

PROGRAM SPECIFIC DATA 3-MAR-82
MICRO2 1L(03) 3-MAR-82 10:02:46
ENKCB Micro-diagnostic for DAP module Rev 01.00

Fiche 1 Frame J9
Sequence 113
Page 107

RETURN
;RETURN LOCATION USED IN TEST #F

```

:4876 .PAGE 'WCS SUBROUTINES FOR CONSOLE USE'
:4877 .TOC  'GENERATE TRANSFER DATA'
:4878 :
:4879 : This routine is used by the diagnostic monitor to load WR 0 with a
:4880 : data pattern from the console. The monitor will set CONSOLE ATTN if
:4881 : a 1 is to be generated, or clear CONSOLE ATTN if a 0 is to be gen-
:4882 : erated. It will then single step this routine to shift the data bit
:4883 : into WR 0. This routine loads a 32 bit value much more quickly than
:4884 : shifting each instruction into the CSR from the monitor, and is used
:4885 : mainly to set up data to load in LS for constants.
:4886 :
:4887 :
:4888 : ***WARNING***
:4889 :
:4890 : This routine must start at 600 and end at 605. DO NOT move this
:4891 : routine to any other area!!
:4892 :
:4893 :
:4894 :
:4895 GEN.XFER.DATA:
:4896 CLP WR[0] ;CLEAR A WORKING REG
:4897 COM WR[0] ;NOW WR 0 IS ALL ONES
:4898 LOOP.XFER:
:4899 SKIP.IF[CONSOLE.ATTN] ;SKIP NEXT INSTRUCTION IS CONSOLE ATTN
:4900 : IS SET (GENERATING A 1)
:4901 ASHL WR[0], ;SHIFT A 0 INTO BIT 0 OF WR 0
:4902 SKIP ;AND SKIP ROL
:4903 ROL WR[0] ;SHIFT A 1 INTO BIT 0 OF WR 0
:4904 JMP [LOOP.XFER] ;REPEAT
:4905 :
:4906 :
:4907 :
:4908 : DIAGNOSTIC VERSION NUMBER IS STORED HERE
:4909 :
:4910 :
:4911 : ***WARNING***
:4912 :
:4913 : These words must be at location 606 and 607. DO NOT move this word
:4914 : to any other area!!
:4915 :
:4916 :
:4917 :
:4918 U 0606, 0001.00 :4918 WORD[0100] ;VERSION 01.00
:4919 U 0607, 0043.42 :4919 ASCII [CB] ;LAST 2 LETTERS OF FILE NAME [ENKCB]
:4920

```

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) DEPOSIT MCT CSR ROUTINE
3-MAR-82 10:02:46

L 9

Fiche 1 Frame L9

Sequence 115
Rev 01.00 Page 109

ENKCB Micro-diagnostic for DAP module

```
:4921 .PAGE  " DEPOSIT MCT CSR ROUTINE"
:4922 .+++
:4923
:4924 .FUNCTIONAL DESCRIPTION:
:4925
:4926     The deposit to CSR routine is used to write the memory controller's
:4927     control and status register. The memory controller has three CSRs
:4928     named CSR 0, CSR 1, and CSR 2.
:4929     CSR 0 contains checkbits or syndrome bits returned from the ECC logic
:4930     after certain diagnostic routines. It is NOT writable directly with
:4931     this routine.
:4932     CSR 1 contains error bits set if an error occurs on a memory access.
:4933     It does contain five maintenance bits (bits 29-25) which are writable.
:4934     There are also seven checkbits (bits 6-0) which are writable, but not
:4935     readable. These bits are used to write checkbits into the ECC chips.
:4936     CSR 2 contains error bits set if a UNIBUS error occurs on a UNIBUS
:4937     access. These are read only bits and are NOT writable by this routine.
:4938     Therefore, CSR 1 is the only writable CSR, and only bits 29-25 can be
:4939     both written and read back. This routine will thus force CSR 1 on a
:4940     deposit to CSR.
:4941     This routine will write LS 5 with data to access CSR 1, then write
:4942     the data residing in LS 6 into the CSR. It will then issue CPU ATTN
:4943     to inform the console that it has completed the function.
:4944     NOTE: The error bits of CSR 1 are cleared on any write to CSR 1.
:4945     These are bits 31, 30 and 23-14. Bits 13-0 and bit 24 are unused.
:4946
:4947 .CALLING SEQUENCE:
:4948
:4949     The console loads the address of a pointer to this routine (a JMP in-
:4950     struction at WCS location 50) into the UPC and starts the CPU.
:4951
:4952 .INPUT PARAMETERS:
:4953
:4954     LS location 6 must have been written by the console with the desired
:4955     data to be deposited in CSR 1 previous to executing this routine.
:4956
:4957 .IMPLICIT INPUTS:
:4958
:4959     None
:4960
:4961 .OUTPUT PARAMATERS:
:4962
:4963     CSR 1 bits 29-25 and 6-0 are written with data from LS 6.
:4964
:4965 .IMPLICIT OUTPUTS:
:4966
:4967     None
:4968
:4969 .SIDE EFFECTS:
:4970
:4971     WR 0 will be modified by this routine.
:4972     LS 5 will be modified by this routine.
:4973     CSR 1 bits 31, 30 and 23-14 are cleared.
:4974
:4975 .---
```

```

:4976
:4977
:4978
U 0608, 3644,15 :4979      MOV LS[CSR1] TO WR[0]      ;SET BIT 2 IN WR 0 AS CSR NUMBER TO
:4980      ; WRITE (BITS 2 AND 3 ARE CSR NUMBER)
U 0609, BE0A,15 :4981      MOV WR[0] TO LS[T5.SUB]      ;PUT CSR NUMBER IN LS LOCATION 5 (CSR
:4982      ; 1)
U 060A, 1D0B,F5 :4983      MEM.REQ[WRITE.CSR] ADRS[T5.SUB] BT[LONG] ;ISSUE MEMORY REQUEST TO
:4984      ; ACCESS CSR 1
U 060B, B20C,15 :4985      WRITE.MEM LS[T6.SUB]      ;SEND DATA IN LS LOCATION 6 TO CSR 1
U 060C, 8868,74 :4986      JMP [WAIT.SUB]      ;JUMP TO SET ATTN AND WAIT
  
```

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

EXAMINE MCT CSR ROUTINE
MICRO2 1L(03) 3-MAR-82 10:02:46
EXAMINE MCT CSR ROUTINE

N 9

Fiche 1 Frame N9

Sequence 117

ENKCB Micro-diagnostic for DAP module Rev 01.00 Page 111

:4987 :PAGE " EXAMINE MCT CSR ROUTINE"

:4988 :+++

:4989

:4990

:4991

:4992

:4993

:4994

:4995

:4996

:4997

:4998

:4999

:5000

:5001

:5002

:5003

:5004

:5005

:5006

:5007

:5008

:5009

:5010

:5011

:5012

:5013

:5014

:5015

:5016

:5017

:5018

:5019

:5020

:5021

:5022

:5023

:5024

:5025

:5026

:5027

:5028

:5029

:5030

:5031

:5032

:5033

:5034

:5035

:5036

:5037

:5038

:5039

:5040

:5041

FUNCTIONAL DESCRIPTION:

The examine CSR routine reads the memory controller's control and status register. There are 3 CSRs labeled CSR 0, CSR 1 and CSR 2.

CSR 0 contains checkbits or syndrome bits from the ECC logic. These are contained in bits 6-0 of the CSR. Other bits are UNPREDICTABLE. Bits 6-0 are only written by special diagnostic routines (they are not writable by the DEPOSIT CSR command).

CSR 1 contains error bits and maintenance bits. The error bits are 31, 30, and 23-14. Maintenance bits are 29-25. The other bits are UNPREDICTABLE. The error bits are written during memory functions only and are not writable via the DEPOSIT CSR command. The maintenance bits are writable. Note that a DEPOSIT CSR command will write the maintenance bits and clear all error bits.

CSR 2 contains three bits (bits 16-14) which are readable. These are set when a UNIBUS error occurs on a UNIBUS access. They are not directly writable with the DEPOSIT CSR command.

The console loads LS 5 with the CSR number (0, 1, or 2) to be read.

This routine rotates LS 5 two places left to position it correctly.

It then executes a read CSR and places the data in LS 6 for the console to read.

CALLING SEQUENCE:

The console loads the address of a pointer to this routine (a JMP instruction at WCS location 51) into the UPC and starts the CPU.

INPUT PARAMETERS:

LS 5 contains the CSR number to be read.

IMPLICIT INPUTS:

None

OUTPUT PARAMETERS:

LS 6 - Data read from CSR selected.

IMPLICIT OUTPUTS:

None

SIDE EFFECTS:

LS 5 is rotated 2 places left.

EXAMINE.CSR:

U 060D, 360A,15

MOV LS[TS.SUB] TO WR[0]

;GET CSR NUMBER FROM LS 5

B 10
Fiche 1 Frame B10
Sequence 118
Page 112

22-ENKCB-1.0 : ENKCB.MIC
: ENKCB.MCR
: ENKCB.MIC

MICRO2 1L(03) 3-MAR-82 10:02:46
EXAMINE MCT CSR ROUTINE

ENKCB Micro-diagnostic for DAP module Rev 01.00

```

J 060E, A2D0,15 :5042      SHL2 WR[0]                ;SHIFT NUMBER 2 PLACES LEFT TO GET
:5043                ;CSR NUMBER IN BITS 2 AND 3
U 060F, BE0A,15 :5044      MOV WR[0] TO LS[T5.SUB]      ;PUT SHIFTED NUMBER IN LS 5
:5045
J 0610, 9D0B,75 :5046      MEM.REQ[READ.CSR] ADRS[T5.SUB] DT[LONG] ;ISSUE MEMORY REQUEST TO
:5047                ; ACCESS CSR SPECIFIED
L 0611, 3022,15 :5048      MOV MEM.DATA TO WR[0]        ;READ CSR SELECTED
:5049
:5050      CHECK.CSR2:
U 0612, B60A,95 :5051      MOV LS[T5.SUB] TO WR[1]      ;GET SHIFTED CSR NUMBER
:5052      CMP LS[#8] WITH WR[1],                      ;SEE IF CSR 2 WAS SELECTED
U 0613, CE46,B5 :5053      DT(LONG)&SET.ALU.CC          ;SET UP CONDITION CODES
U 0614, 0861,61 :5054      JMP.IF[NEQ] TO [CHECK.CSR1]   ;JUMP IF CSR 2 WAS NOT SELECTED
U 0615, C530,15 :5055      BIC LS[CSR2.MASK] TO WR[0]    ;CLEAR UNUSED (UNPREDICTABLE) BITS IN
:5056                ; CSR 2.
:5057      CHECK.CSR1:
:5058      CMP LS[#4] WITH WR[1],                      ;SEE IF CSR 1 WAS SELECTED
U 0616, 4E44,B5 :5059      DT(LONG)&SET.ALU.CC          ;SET UP CONDITION CODES
U 0617, 0861,91 :5060      JMP.IF[NEQ] TO [CHECK.CSR0]   ;JUMP IF CSR 1 WAS NOT SELECTED
U 0618, C52E,15 :5061      BIC LS[CSR1.MASK] TO WR[0]    ;CLEAR UNUSED (UNPREDICTABLE) BITS IN
:5062                ; CSR 1.
:5063      CHECK.CSR0:
:5064      CMP LS[#0] WITH WR[1],                      ;SEE IF CSR 0 WAS SELECTED
U 0619, 4E9C,B5 :5065      DT(LONG)&SET.ALU.CC          ;SET UP CONDITION CODES
U 061A, 8861,D1 :5066      JMP.IF[NEQ] TO [LOAD.LS6]     ;JUMP IF CSR 0 WAS NOT SELECTED
U 061B, 452C,15 :5067      BIC LS[CSR0.MASK] TO WR[0]    ;CLEAR UNUSED (UNPREDICTABLE) BITS IN
:5068                ; CSR 0.
U 061C, C54E,15 :5069      BIC LS[BIT7] TO WR[0]        ;BIT 7 IS ALSO UNUSED
:5070
:5071      LOAD.LS6:
U 061D, BE0C,15 :5072      MOV WR[0] TO LS[T6.SUB]      ;LOAD CSR DATA INTO LS 6 FOR PRINTOUT
U 061E, 8868,74 :5073      JMP [WAIT.SUB]              ;ISSUE CPU ATTN AND WAIT FOR RESPONSE

```


ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) 3-MAR-82 10:02:46
DEPOSIT MCT TRANSLATION BUFFER ROUTINE

C 10

Fiche 1 Frame C10

Sequence 119

ENKCB Micro-diagnostic for DAP module Rev 01.00 Page 113

:5074 :page " DEPOSIT MCT TRANSLATION BUFFER ROUTINE"
:5075 :+++
:5076

:5077 :FUNCTIONAL DESCRIPTION:
:5078

:5079 : This routine allows a write to the translation buffer portion of the
:5080 : memory controller. The buffer area contains 128 decimal locations
:5081 : addressed from 0-7F(H). The remainder of the TB RAM is either unused
:5082 : or contains the UNIBUS MAP. The DEPOSIT UB command is used to write
:5083 : the UNIBUS map portion of the TB RAMs. The address specified with the
:5084 : DEPOSIT command will be the actual RAM address accessed. This routine
:5085 : will do any shifting necessary on the address specified to position it
:5086 : correctly. The address specified has been loaded by the console in LS
:5087 : location 5 prior to executing this routine. The address must be in HEX
:5088 : format.

:5089 : The data specified has been placed in LS 6 by the console prior to
:5090 : executing this routine. Bits 07-01 will be the Valid, Prot, Modify,
:5091 : and Byte Offset bits (there are four Prot bits). Bits 22-08 will
:5092 : contain the PA bits from PA 23 through PA 09 respectively. Bits 31
:5093 : through 23 and bit 0 are unused. This routine will do any shifting nec-
:5094 : essary to write the bits in TB as described. The data specified must
:5095 : be in HEX format.
:5096

:5097 :CALLING SEQUENCE:
:5098

:5099 : The console loads the address of a pointer to this routine (a JMP in-
:5100 : struction at WCS location 52) into the UPC and starts the CPU.
:5101

:5102 :INPUT PARAMETERS:
:5103

:5104 : LS 5 - TB address to be written (range 0-7F)
:5105 : LS 6 - TB data to write (22 bits, from 1-22). Bits 0 and 23-31 are
:5106 : ignored.
:5107

:5108 :IMPLICIT INPUTS:
:5109

:5110 : None
:5111

:5112 :OUTPUT PARAMETERS:
:5113

:5114 : TB will be written with specified data at specified address
:5115

:5116 :IMPLICIT OUTPUTS:
:5117

:5118 : None
:5119

:5120 :SIDE EFFECTS:
:5121

:5122 : WR 0 will be modified
:5123 : WR 1 will be modified
:5124

:5125 :---
:5126

:5127 :DEPOSIT.TB:
:5128

JSR [SHIFT.TB.ADR]

:GO TO SUBROUTINE TO POSITION TB AD-

U 061F, 8862,FC

```

:5129
U 0620, 360C,15 :5130      MOV LS[T6.SUB] TO WR[0]      : DRESS CORRECTLY
U 0621, 4526,15 :5131      BIC LS[FFFF] TO WR[0]      : GET DATA FROM LS 6
:5132      : CLEAR LOW BYTE OF DATA SPECIFIED.
:5133      : LOW BYTE WILL BE PUT IN BITS 24-31
:5134      : LATER
U 0622, 8862,AC :5135      JSR [SHIFT.LOOP]      : GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
:5136      : PLACES RIGHT
U 0623, E40C,15 :5137      SWAP LS[T6.SUB] WITH WR[0] : SWAP ORIGINAL DATA SPECIFIED WITH MOD-
:5138      : IFIED DATA. NOW LS 6 CONTAINS BITS
:5139      : 8-22 IN BITS 0-14 AS REQUIRED. WR 0
:5140      : CONTAINS ORIGINAL DATA SPECIFIED
U 0624, 452C,15 :5141      BIC LS[FFFFFF00] TO WR[0] : CLEAR ALL BUT LOW BYTE OF ORIGINAL
:5142      : DATA SPECIFIED
U 0625, 8862,AC :5143      JSR [SHIFT.LOOP]      : GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
:5144      : PLACES RIGHT. ON RETURN, WR 0 WILL
:5145      : HAVE BITS 0-7 IN BITS 24-31, OTHER
:5146      : BITS ARE CLEAR
U 0626, ED0C,15 :5147      BIS WR[0] TO LS[T6.SUB] : NOW LS 6 CONTAINS DATA TO WRITE IN
:5148      : TB IN CORRECT FORMAT I.E. PA BITS
:5149      : IN BITS 00-14 AND BYTE OFFSET, MODIFY
:5150      : PROT A THROUGH PROT D, AND VALID IN
:5151      : BITS 25-31.
U 0627, 980A,F5 :5152      MEM.REQ[WRITE.TB] ADRS[T5.SUB] DT[LONG] :ISSUE MEMORY REQUEST TO WRITE
:5153      : THE TB
U 0628, B20C,15 :5154      WRITE.MEM LS[T6.SUB]      :WRITE DATA FROM LS 6 IN TB
U 0629, 8868,74 :5155      JMP [WAIT.SUB]          :JUMP TO SET ATTN AND WAIT
:5156
:5157
:5158      SUBROUTINE TO SHIFT WR 0 TO THE RIGHT 8 PLACES
:5159
:5160      SHIFT.LOOP:
U 062A, 3646,95 :5161      MOV LS[8] TO WR[1]      :COUNTER IN WR 1 TO SHIFT 8 PLACES
:5162      SHIFT.LOOP.1:
U 062B, 2340,15 :5163      ROR WR[0]              :SHIFT WR 0 ONE PLACE RIGHT
:5164      DEC WR[1]              :DECREMENT COUNTER
U 062C, A102,B5 :5165      DT[LONG]&SET.ALU.CC      : AND SET CONDITION CODES
U 062D, 8862,B1 :5166      JMP.IF[NEQ] TO [SHIFT.LOOP.1] :REPEAT UNTIL 8 SHIFTS HAVE OCCURRED
U 062E, 5B00,14 :5167      RETURN              :THEN RETURN
:5168
:5169      SUBROUTINE TO POSITION TB ADDRESS IN BITS 9-14 AND BIT 31 (TB ADDRESS
:5170      BIT 0 MUST BE IN BIT 31).
:5171
:5172      SHIFT.TB.ADR:
U 062F, 360A,15 :5173      MOV LS[T5.SUB] TO WR[0]      :GET 1B ADDRESS FROM LS 5
U 0630, 452C,15 :5174      BIC LS[FFFFFF00] TO WR[0] :CLEAR ALL BUT LOW BYTE (8 BIT ADDRESS)
U 0631, A2D0,15 :5175      SHL2 WR[0]              :SHIFT 2 PLACES LEFT TO POSITION ADDRESS
U 0632, A2D0,15 :5176      SHL2 WR[0]              :SHIFT 2 PLACES LEFT TO POSITION ADDRESS
U 0633, A2D0,15 :5177      SHL2 WR[0]              :SHIFT 2 PLACES LEFT TO POSITION ADDRESS
U 0634, A2D0,15 :5178      SHL2 WR[0]              :SHIFT 2 PLACES LEFT TO POSITION ADDRESS
U 0635, 23D0,15 :5179      ASHL WR[0]              :SHIFT ONE MORE PLACE
:5180      : NOW BITS 0-6 IN BITS 9-15
:5181      : SEE IF BIT 15 (MSB) IS SET OR CLEAR.
U 0636, 595E,35 :5182      BIT LS[BIT15] WITH WR[0], : MSB MUST GO IN BIT 31 IN ORDER TO AD-
:5183      DT[LONG]&SET.ALU.CC      : DRESS TB CORRECTLY

```

ZZ-ENKCB-1.0	:	ENKCB.MIC							
:	:	ENKCB.MCR	MICRO2 1L(03)	3-MAR-82	10:02:46	ENKCB Micro-diagnostic for DAP module	Rev 01.00	Page 115	
:	:	ENKCB.MIC	DEPOSIT MCT TRANSLATION BUFFER ROUTINE						
U 0637, 5B00,09	:	5184	SKIP.IF[EQ]			;SKIP NEXT INSTRUCTION IF MSB IS 0			
U 0638, 477E,15	:	5185	BIS LS[BIT31] TO WR[0]			;ELSE SET BIT 31 FOR MSB			
	:	5186							
	:	5187	MOV WR[0] TO LS[TS.SUB],			;NOW TB ADDRESS IS IN CORRECT FORM IN			
	:	5188				; LS LOCATION 5			
U 0639, 3E0A,14	:	5189	RETURN			; THEN RETURN TO MAIN CODE			

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) EXAMINE TRANSLATION 3-MAR-82
EXAMINE TRANSLATION BUFFER ROUTINE

F 10

Fiche 1 Frame F10

Sequence 122

ENKCB Micro-diagnostic for DAP module Rev 01.00 Page 116

```
:5190 .PAGE  " EXAMINE TRANSLATION BUFFER ROUTINE"
:5191 .+++
:5192
:5193 .FUNCTIONAL DESCRIPTION:
:5194
:5195 .This routine will read the translation buffer of the memory controller.
:5196 .The TB will be returned with the byte offset, modify, prot A through
:5197 .prot D, and valid in bits 01-07 respectively. PA bits 09-23 are re-
:5198 .turned in bits 08-22 respectively. Bits 23-31 and bit 0 are not used
:5199 .and so will be cleared before printing the result. The result will be
:5200 .put in LS 6 for the console to print.
:5201 .The routine will issue a memory request with memory function=READ.TB
:5202 .and data type=LONGWORD. The console will load LS 5 with the address
:5203 .specified before executing this routine. The routine will shift the
:5204 .address around as required to address the TB location specified. The
:5205 .address specified must be in HEX format.
:5206 .The TB consists of 128 decimal locations (addresses from 0-7F). The
:5207 .remaining locations in the TB RAM are either unused or used by the
:5208 .UNIBUS map. UNIBUS map addresses are read via the EXAMINE UB routine.
:5209
:5210 .CALLING SEQUENCE:
:5211
:5212 .The console loads the address of a pointer to this routine (a JMP in-
:5213 .struction at WCS location 53) into the UPC and starts the CPU.
:5214
:5215 .INPUT PARAMETERS:
:5216
:5217 .LS 5 - TB address (0-7F) in hex format
:5218
:5219 .IMPLICIT INPUTS:
:5220
:5221 .None
:5222
:5223 .OUTPUT PARAMATERS:
:5224
:5225 .LS 6 - TB data from address specified
:5226
:5227 .IMPLICIT OUTPUTS:
:5228
:5229 .None
:5230
:5231 .SIDE EFFECTS:
:5232
:5233 .LS 5 is modified
:5234 .WR 0 is modified
:5235
:5236 .---
:5237
:5238 .EXAMINE.TB:
:5239 .JSR [SHIFT.TB.ADR] ;SHIFT ADDRESS SPECIFIED FROM LS 5 INTO
:5240 . ; CORRECT POSITION AND REPLACE IN LS 5
:5241 .MEM.REQ[READ.TB] ADRS[T5.SUB] DT[LONG] ;ISSUE MEMORY REQUEST TO READ
:5242 . ; THE TB AT ADDRESS CONTAINED IN LS 5
:5243 .MOV MEM.DATA TO WR[0] ;PLACE RESULT IN WR 0
:5244
```

U 063A, 8862,FC

U 063B, 9C0B,F5

U 063C, 3022,15

```

U 063D, 452A,15 :5245      BIC LS[0] TO WR[0]      ;CLEAR UPPER BYTE OF RESULT (UPPER BYTE
                    :5246      ; IS NOT PART OF TB AND IS UNPREDICT-
                    :5247      ; ABLE)
U 063E, 456E,15 :5248      BIC LS[BIT23] TO WR[0]      ;DITTO FOR BIT 23
U 063F, 4540,15 :5249      BIC LS[BIT0] TO WR[0]      ;DITTO FOR BIT 0.  WR 0 NOW CONTAINS
                    :5250      ; RESULT TO PRINT
L 0640, 8E0C,15 :5251      MOV WR[0] TO LS[T6.SUB]      ;RESULT NOW IN LS 6
U 0641, 8868,74 :5252      JMP [WAIT.SUB]              ;JUMP TO SET ATTN AND WAIT

```

```

:5253 .PAGE  '' DEPOSIT UNIBUS MAP ROUTINE''
:5254 .+++
:5255
:5256 .FUNCTIONAL DESCRIPTION:
:5257
:5258 .    This routine allows a write to the UNIBUS map portion of the TB on the
:5259 .    memory controller.  The map area contains 512 decimal locations
:5260 .    addressed from 200-3FF.  The remainder of the TB RAM is either unused
:5261 .    or contains the Translation Buffer information.  The DEPOSIT TB command
:5262 .    is used to write the TB portion of the TB RAMs.  The address specified
:5263 .    with the DEPOSIT command will be the actual RAM address accessed.  This
:5264 .    routine will do any shifting necessary on the address specified to pos-
:5265 .    ition it correctly.  The address specified has been loaded by the con-
:5266 .    sole in LS location 5 prior to executing this routine.  The address
:5267 .    must be in HEX format.
:5268 .    The data specified has been placed in LS 6 by the console prior to
:5269 .    executing this routine.  Bits 07-01 will be the Valid, Prot, Modify,
:5270 .    and Byte Offset bits (there are four Prot bits).  Bits 22-08 will
:5271 .    contain the PA bits from PA 23 through PA 09 respectively.  Bits 31
:5272 .    through 23 and bit 0 are unused.  This routine will do any shifting nec-
:5273 .    essary to write the bits in TB as described.  The data specified must
:5274 .    be in HEX format.
:5275
:5276 .CALLING SEQUENCE:
:5277
:5278 .    The console loads the address of a pointer to this routine (a JMP in-
:5279 .    struction at WCS location 58) into the UPC and starts the CPU.
:5280
:5281 .INPUT PARAMETERS:
:5282
:5283 .    LS 5 - UNIBUS Map address in TB (must be in range 200-3FF HEX).
:5284 .    LS 6 - Contains the data to write into the UNIBUS Map in bits 1-22.
:5285
:5286 .IMPLICIT INPUTS:
:5287
:5288 .    None
:5289
:5290 .OUTPUT PARAMATERS:
:5291
:5292 .    The UNIBUS map portion of the TB RAMs gets written with the data from
:5293 .    LS 6 at the address specified in LS 5.
:5294
:5295 .IMPLICIT OUTPUTS:
:5296
:5297 .    None
:5298
:5299 .SIDE EFFECTS:
:5300
:5301 .    WR 0 will be modified
:5302 .    WR 1 will be modified
:5303
:5304 .---
:5305
:5306 .DEPOSIT.UBS:
:5307 .    JSR [SHIFT.UB.ADR]                ;GO TO SUBROUTINE TO ARRANGE UNIBUS
  
```

```

:5308
U 0643, 360C,15 :5309      MOV LS[T6.SUB] TO WR[0]      : MAP ADDRESS CORRECTLY
U 0644, 4526,15 :5310      BIC LS[0FF] TO WR[0]      : GET DATA FROM LS 6
:5311      : CLEAR LOW BYTE OF DATA SPECIFIED.
:5312      : LOW BYTE WILL BE PUT IN BITS 24-31
:5313      : LATER
U 0645, 8862,AC :5314      JSR [SHIFT.LOOP]      : GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
:5315      : PLACES RIGHT
U 0646, E40C,15 :5316      SWAP LS[T6.SUB] WITH WR[0]      : SWAP ORIGINAL DATA SPECIFIED WITH MOD-
:5317      : IFIED DATA. NOW LS 6 CONTAINS BITS
:5318      : 8-22 IN BITS 0-14 AS REQUIRED. WR 0
:5319      : CONTAINS ORIGINAL DATA SPECIFIED
U 0647, 452C,15 :5320      BIC LS[0FFFFFF00] TO WR[0]      : CLEAR ALL BUT LOW BYTE OF ORIGINAL
:5321      : DATA SPECIFIED
U 0648, 8862,AC :5322      JSR [SHIFT.LOOP]      : GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
:5323      : PLACES RIGHT. ON RETURN, WR 0 WILL
:5324      : HAVE BITS 0-7 IN BITS 24-31, OTHER
:5325      : BITS ARE CLEAR
U 0649, E00C,15 :5326      BIS WR[0] TO LS[T6.SUB]      : NOW LS 6 CONTAINS DATA TO WRITE IN
:5327      : TB IN CORRECT FORMAT I.E. PA BITS
:5328      : IN BITS 00-14 AND BYTE OFFSET, MODIFY
:5329      : PROT A THROUGH PROT D, AND VALID IN
:5330      : BITS 25-31.
U 064A, 1B0B,F5 :5331      MEM.REQ[WRITE.UBS.MAP] ADRS[T5.SUB] DT[LONG] :ISSUE MEMORY REQUEST TO
:5332      : WRITE THE UNIBUS MAP
U 064B, B20C,15 :5333      WRITE.MEM LS[T6.SUB]      :WRITE DATA FROM LS 6 IN UNIBUS MAP
U 064C, 8868,74 :5334      JMP [WAIT.SUB]      :JUMP TO SET ATTN AND WAIT
:5335      :
:5336      : SUBROUTINE TO POSITION ADDRESS SPECIFIED CORRECTLY TO ADDRESS THE
:5337      : UNIBUS MAP.
:5338      :
:5339      : SHIFT.UB.ADR:
U 064D, 360A,15 :5340      MOV LS[T5.SUB] TO WR[0]      : GET ADDRESS SPECIFIED IN WR 0
U 064E, A2D0,15 :5341      SHL2 WR[0]      :SHIFT ADDRESS 2 PLACES LEFT
U 064F, A2D0,15 :5342      SHL2 WR[0]      :SHIFT ADDRESS 2 PLACES LEFT
U 0650, A2D0,15 :5343      SHL2 WR[0]      :SHIFT ADDRESS 2 PLACES LEFT
U 0651, A2D0,15 :5344      SHL2 WR[0]      :SHIFT ADDRESS 2 PLACES LEFT
U 0652, 23D0,15 :5345      ASHL WR[0]      :SHIFT LEFT ONE MORE PLACE
:5346      : NOW ADDRESS IS IN BITS 9-17 OF WR 0
:5347      : NOW LS 5 CONTAINS CORRECT ADDRESS.
U 0653, 3E0A,14 :5348      MOV WR[0] TO LS[T5.SUB],
:5349      : RETURN
:5350      : RETURN TO MAIN

```

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

J 10
EXAMINE UNIBUS MAP 3-MAR-82
MICRO2 1L(03) 3-MAR-82 10:02:46
EXAMINE UNIBUS MAP ROUTINE
Fiche 1 Frame J10
ENKCB Micro-diagnostic for DAP module
Sequence 126
Rev 01.00
Page 120

```
:5351 .PAGE  " EXAMINE UNIBUS MAP ROUTINE"
:5352 .+++
:5353 .
:5354 .   This routine will read the UNIBUS map portion of the memory controller.
:5355 .   The UNIBUS MAP contents will be returned with the byte offset, modify,
:5356 .   prot A through prot D, and valid in bits 01-07 respectively. PA bits
:5357 .   09-23 are returned in bits 08-22 respectively. Bits 23-31 and bit 0
:5358 .   are not used and so will be cleared before printing the result. The
:5359 .   result will be put in LS 6 for the console to print.
:5360 .   The routine will issue a memory request with memory function=READ.UBS
:5361 .   .MAP and data type=LONGWORD. The console will load LS 5 with the ad-
:5362 .   dress specified before executing this routine. The routine will shift
:5363 .   the address around as required to address the UNIBUS map location spec-
:5364 .   ified. The address specified must be in HEX format.
:5365 .   The UNIBUS map consists of 512 decimal locations (addresses from 200-
:5366 .   3FF) The remaining locations in the TB RAM are either unused or used
:5367 .   by the translation buffer. TB contents are read via the EXAMINE TB
:5368 .   routine.
:5369 .
:5370 . CALLING SEQUENCE:
:5371 .
:5372 .   The console loads the address of a pointer to this routine (a JMP in-
:5373 .   struction at WCS location 59) into the UPC and starts the CPU.
:5374 .
:5375 . INPUT PARAMETERS:
:5376 .
:5377 .   LS 5 - TB address in hex format
:5378 .
:5379 . IMPLICIT INPUTS:
:5380 .
:5381 .   None
:5382 .
:5383 . OUTPUT PARAMATERS:
:5384 .
:5385 .   LS 6 - UNIBUS map data from address specified
:5386 .
:5387 . IMPLICIT OUTPUTS:
:5388 .
:5389 .   None
:5390 .
:5391 . SIDE EFFECTS:
:5392 .
:5393 .   LS 5 is modified
:5394 .   WR 0 will be modified
:5395 .
:5396 . ---
:5397 . EXAMINE.UBS:
:5398 .   JSR [SHIFT.UB.ADR]           ;SHIFT ADDRESS SPECIFIED FROM LS 5 INTO
:5399 .                               ; CORRECT POSITION AND REPLACE IN LS 5
:5400 .   MEM.REQ[READ.UBS.MAP] ADRS[TS.SUB] DT[LONG] ;ISSUE MEMORY REQUEST TO
:5401 .                               ; READ THE UBS MAP AT ADDRESS CONTAINED
:5402 .                               ; IN LS 5
:5403 .   MOV MEM.DATA TO WR[0]        ;PLACE RESULT IN WR 0
:5404 .
:5405 .   BIC LS[0FF000000] TO WR[0]   ;CLEAR UPPER BYTE OF RESULT (UPPER BYTE
```

U 0654, 0864,DC

U 0655, 1C0B,75

U 0656, 3022,15

U 0657, 452A,15

U 0658, 456E,15	:5406		: IS NOT PART OF TB AND IS UNPREDICT-
U 0659, 4540,15	:5407		: ABLE)
	:5408	BIC LS[BIT23] TO WR[0]	:DITTO FOR BIT 23
U 065A, BE0C,15	:5409	BIC LS[BIT0] TO WR[0]	:DITTO FOR BIT 0. WR 0 NOW CONTAINS
L 065B, 8868,74	:5410		: RESULT TO PRINT
	:5411	MOV WR[0] TO LS[16.SUB]	:RESULT NOW IN LS 6
	:5412	JMP [WAIT.SUB]	:JUMP TO SET ATTN AND WAIT
	:5413		: CONTROL

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) 3-MAR-82 10:02:46
DEPOSIT MAIN MEMORY ROUTINE

L 10

Fiche 1 Frame L10

Sequence 128

ENKCB Micro-diagnostic for DAP module Rev 01.00 Page 122

```
:5414 .PAGE  " DEPOSIT MAIN MEMORY ROUTINE"
:5415 .+++
:5416
:5417 .FUNCTIONAL DESCRIPTION:
:5418
:5419 .    This routine write a longword of data into main memory at the Physical
:5420 .    address specified. This address need not be on a longword boundary.
:5421 .    The console will use this routine for data transfers to main memory as
:5422 .    well as for the DEPOSIT MM command. The console will have loaded the
:5423 .    address into LS 5 and the data into LS 6 before executing this routine.
:5424 .    The routine issues a memory request with the memory function=WRITE.P
:5425 .    and the data type=longword. It then issues a WRITE.MEM LS[6] instruc-
:5426 .    tion to write the data into memory. A check on ERROR SUM is made to
:5427 .    assure that the write occurred without an error. If ERROR SUM is as-
:5428 .    serted, the CPU will issue a CPU ACK before asserting CPU ATTN to in-
:5429 .    form the console that an error occurred.
:5430
:5431 .CALLING SEQUENCE:
:5432
:5433 .    The console loads the address of a pointer to this routine (a JMP in-
:5434 .    struction at WCS location 54) into the UPC and starts the CPU.
:5435
:5436 .INPUT PARAMETERS:
:5437
:5438 .    LS 5 - Main memory physical address
:5439 .    LS 6 - Main memory longword data
:5440
:5441 .IMPLICIT INPUTS:
:5442
:5443 .    None
:5444
:5445 .OUTPUT PARAMATERS:
:5446
:5447 .    Main Memory address specified is written with data from LS 6.
:5448
:5449 .IMPLICIT OUTPUTS:
:5450
:5451 .    A memory error asserts CPU ACK to inform the console of a write error.
:5452
:5453 .SIDE EFFECTS:
:5454
:5455 .    None
:5456
:5457 .---
:5458
:5459 .DEPOSIT.MM:
:5460 .    MEM.REQ[WRITE.P] ADRS[T5.SUB] DT[LONG] ;SET UP FOR WRITE TO MAIN
:5461 .    ; MEMORY USING THE PHYSICAL ADDRESS
:5462 .    ; STORED IN LS 5
:5463 .    WRITE.MEM LS[T6.SUB], ;WRITE THE DATA FROM LS 6 INTO MAIN
:5464 .    ; MEMORY
:5465 .    SKIP.IF[MEM.REF.OK] ;SKIP NEXT INSTRUCTION IF NO MEMORY
:5466 .    ; ERROR (NO ERR SUM)
:5467 .    MISC [SET.CP.ACK] ;SET CPU ACK IF ERROR SUM ASSERTED
:5468
```

U 065C, 990A,75

U 065D, 320C,1D

U 065E, 10D0,15


```

:5472 .PAGE  " EXAMINE MAIN MEMORY ROUTINE"
:5473 .+++
:5474
:5475 .FUNCTIONAL DESCRIPTION:
:5476
:5477     This routine is used to read a longword form main memory at the Phy-
:5478     sical address specified. The address need not be on a longword bound-
:5479     ary. This routine is used by the console on an EXAMINE.MM command.
:5480     The console will have loaded LS 5 with the physical address specified
:5481     before executing this routine.
:5482     The routine will first write CSR 1 to set the INH REP CRD (inhibit
:5483     report correctable read data) bit to prevent an error sum from occurring
:5484     on a single bit error which has been corrected. Then the routine is-
:5485     sues a memory request with memory function=READ.P and DT=longword. It
:5486     then reads the memory location and puts the result in LS 6. It also
:5487     checks for an ERROR SUM and issues a CPU ACK if an error occurred (sin-
:5488     gle bit errors that have been corrected will not cause an ERR SUM). If
:5489     an error occurred, CPU ACK will be set before issueing CPU ATTN to in-
:5490     form the console that an error occurred.
:5491
:5492 .CALLING SEQUENCE:
:5493
:5494     The console loads the address of a pointer to this routine (a JMP in-
:5495     struction at WCS location 55) into the UPC and starts the CPU.
:5496
:5497 .INPUT PARAMETERS:
:5498
:5499     LS 5 - Physical address in main memory to read.
:5500
:5501 .IMPLICIT INPUTS:
:5502
:5503     LS[INH.CRD] - Data to write into CSR 1 to inhibit error sum on a CRD.
:5504     LS[CSR1] - Address for writing CSR 1.
:5505
:5506 .OUTPUT PARAMATERS:
:5507
:5508     LS 6 - Longword data from physical memory address specified.
:5509
:5510 .IMPLICIT OUTPUTS:
:5511
:5512     CPU ACK if an error sum occurs.
:5513
:5514 .SIDE EFFECTS:
:5515
:5516     None
:5517
:5518 .---
:5519
:5520 EXAMINE.MM:
:5521     MEM.REQ[WRITE.CSR] ADRS[CSR1] DT[LONG] ;ISSUE MEMORY REQUEST TO
:5522     ; WRITE CSR 1
:5523     WRITE.MEM LS[INH.CRD] ;SET INH REP CRD IN CSR 1 AND CLEAR
:5524     ; ECC DISABLE
:5525
:5526     MEM.REQ[READ.P] ADRS[TS.SUB] DT[LONG] ;ISSUE MEMORY REQUEST TO
  
```

U 0662, 1045,F5

U 0663, B278,15

U 0664, 180B,F5

llll

:5534 .page " EXAMINE IDC ROUTINES"

:5535 :++

:5536 :
:5537 : FUNCTIONAL DESCRIPTION:

:5538 :
:5539 : The following routines are used to read data from the optional IDC
:5540 : module's registers. The result is put in LS 6 for the console to
:5541 : print out.

:5542 :
:5543 : CALLING SEQUENCE:

:5544 :
:5545 : The console loads the address of a pointer to this routine (a JMP in-
:5546 : struction at WCS location 5A through 5E) into the UPC and starts the
:5547 : CPU.

:5548 :
:5549 : INPUT PARAMETERS:

:5550 :
:5551 : None

:5552 :
:5553 : IMPLICIT INPUTS:

:5554 :
:5555 : None

:5556 :
:5557 : OUTPUT PARAMATERS:

:5558 :
:5559 : LS 6 - Data from IDC register specified.

:5560 :
:5561 : IMPLICIT OUTPUTS:

:5562 :
:5563 : None

:5564 :
:5565 : SIDE EFFECTS:

:5566 :
:5567 : None

:5568 :
:5569 : ---

:5570 :
:5571 : EXAMINE.ICSR:

U 0668, 9090,15	:5572	MISC [SET.PORT.I.0]	:SELECT PORT TO BUS
U 0669, 9780,15	:5573	PORT.READ PORT.ADRS[CSR]	:SEND THE READ COMMAND
U 066A, 0867,64	:5574	JMP [READ.IDC]	:READ THE DATA

:5575 :
:5576 : EXAMINE.IDAR:

U 066B, 9090,15	:5577	MISC [SET.PORT.I.0]	:SELECT PORT TO BUS
U 066C, 1784,15	:5578	PORT.READ PORT.ADRS[DAR]	:SEND THE READ COMMAND
U 066D, 0867,64	:5579	JMP [READ.IDC]	:READ THE DATA

:5580 :
:5581 : EXAMINE.DBUF:

U 066E, 9090,15	:5582	MISC [SET.PORT.I.0]	:SELECT PORT TO BUS
U 066F, 978C,15	:5583	PORT.READ PORT.ADRS[DATA.WORD]	:SEND THE READ COMMAND
U 0670, 0867,64	:5584	JMP [READ.IDC]	:READ THE DATA

:5585 :
:5586 : EXAMINE.PATT:

U 0671, 9090,15	:5587	MISC [SET.PORT.I.0]	:SELECT PORT TO BUS
U 0672, 1790,15	:5588	PORT.READ PORT.ADRS[PATTERN]	:SEND THE READ COMMAND

D 11
Fiche 1 Frame D11
Sequence 133
Page 127

ZZ-ENKCB-1.0 : ENKCB.MIC
 : ENKCB.MCR
 : ENKCB.MIC

MICRO2 1L(03) 3-MAR-82 10:02:46
 EXAMINE IDC ROUTINES

U 0673, 0867,64 :5589 JMP [READ.IDC] ;READ THE DATA
 :5590
 :5591 EXAMINE.POSIT:
 U 0674, 9090,15 :5592 MISC [SET.PORT.1.0] ;SELECT PORT TO BUS
 U 0675, 979 15 :5593 PORT.READ PORT.ADRS[POSITION] ;SEND THE READ COMMAND
 :5594
 :5595
 :5596 READ.IDC:
 U 0676, 3A0C,15 :5597 MOV PORT TO LS[T6.SUB] ;READ DATA AND PUT IN LS 6
 U 0677, 1080,15 :5598 MISC [SET.ACC.1.0] ;RETURN SELECT TO ACCELERATOR
 U 0678, 8868,74 :5599 JMP [WAIT.SUB] ;JUMP TO SET ATTN AND WAIT

```

:5600 .page  " DEPOSIT IDC ROUTINES"
:5601 .+++
:5602
:5603 .FUNCTIONAL DESCRIPTION:
:5604
:5605 .The following routines are used to write data to the optional IDC
:5606 .module's registers. The data is taken from LS 6 which is previously
:5607 .loaded by the monitor when the DEPOSIT command is executed.
:5608
:5609 .CALLING SEQUENCE:
:5610
:5611 .The console loads the address of a pointer to this routine (a JMP in-
:5612 .struction at WCS location 5F through 61) into the UPC and starts the
:5613 .CPU.
:5614
:5615 .INPUT PARAMETERS:
:5616
:5617 .LS 6 - Data pattern to write.
:5618
:5619 .IMPLICIT INPUTS:
:5620
:5621 .None
:5622
:5623 .OUTPUT PARAMETERS:
:5624
:5625 .None
:5626
:5627 .IMPLICIT OUTPUTS:
:5628
:5629 .None
:5630
:5631 .SIDE EFFECTS:
:5632
:5633 .None
:5634
:5635 .---
:5636
:5637 .DEPOSIT.ICSR:
U 0679, 360C,15 :5638 .MOV LS[T6.SUB] TO WR[0] ;GET DATA PATTERN TO WRITE
U 067A, 17A0,15 :5639 .PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE TO IDC
U 067B, 8868,74 :5640 .JMP [WAIT.SUB] ;JUMP TO SET ATTN AND WAIT
:5641
:5642
:5643 .DEPOSIT.IDAR:
U 067C, 360C,15 :5644 .MOV LS[T6.SUB] TO WR[0] ;GET DATA PATTERN TO WRITE
U 067D, 97A4,15 :5645 .PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;WRITE TO IDC
U 067E, 8868,74 :5646 .JMP [WAIT.SUB] ;JUMP TO SET ATTN AND WAIT
:5647
:5648
:5649 .DEPOSIT.DBUF:
U 067F, 360C,15 :5650 .MOV LS[T6.SUB] TO WR[0] ;GET DATA PATTERN TO WRITE
U 0680, 17AC,15 :5651 .PORT.WRITE PORT.ADRS[DATA.WORD] WITH WR[0] ;WRITE TO IDC
U 0681, 8868,74 :5652 .JMP [WAIT.SUB] ;JUMP TO SET ATTN AND WAIT
:5653
:5654

```



```

:5655 CLEAR.FIFO.ADDR:
U 0682, 17C0,15 :5656 PORT,CONTROL [CLEAR.FIFO.CNTR] ;CLEAR ADDRESS IN FIFO A OR FIFO B
U 0683, 8868,74 :5657 JMP [WAIT.SUB] ;JUMP TO SET ATTN AND WAIT
:5658
:5659
:5660 SELECT.FIFO.A:
L 0684, 17D8,15 :5661 PORT,CONTROL [SEL.FIFO.A] ;SELECT FIFO A
U 0685, 8868,74 :5662 JMP [WAIT.SUB] ;JUMP TO SET ATTN AND WAIT
:5663
:5664
:5665 SELECT.FIFO.B:
U 0686, 97DC,15 :5666 PORT,CONTROL [SEL.FIFO.B] ;SELECT FIFO B
:5667
:5668 WAIT.SUB:
U 0687, 10E0,15 :5669 MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:5670 WAIT.DE.EX:
U 0688, 8868,84 :5671 JMP [WAIT.DE.EX] ;LOOP SELF UNTIL CONSOLE TAKES CONTROL
  
```

```

:5672 .PAGE  " SAVE WR ROUTINE"
:5673 .+++
:5674
:5675 .FUNCTIONAL DESCRIPTION:
:5676
:5677 .    This routine saves WRs 0-3 in LS locations 1-4.  When the console takes
:5678 .    control of the CPU, it calls this routine so that the working registers
:5679 .    can be used by the console.  The WRs are restored by another routine
:5680 .    called by the console before the CPU resumes execution (on a COnfigure
:5681 .    command, for example).
:5682 .    This routine simply moves data from WR 0-3 to LS locations 1-4 and
:5683 .    then issues a CPU ATTN to the console.  It then loops on a JMP self
:5684 .    until the console takes over.
:5685
:5686 .CALLING SEQUENCE:
:5687
:5688 .    The console loads the address of a pointer to this routine (a JMP in-
:5689 .    struction at WCS location 46) into the UPC and starts the CPU.
:5690
:5691 .INPUT PARAMETERS:
:5692
:5693 .    None
:5694
:5695 .IMPLICIT INPUTS:
:5696
:5697 .    None
:5698
:5699 .OUTPUT PARAMATERS:
:5700
:5701 .    LS 1 - Contents of WR 0
:5702 .    LS 2 - Contents of WR 1
:5703 .    LS 3 - Contents of WR 2
:5704 .    LS 4 - Contents of WR 3
:5705
:5706 .IMPLICIT OUTPUTS:
:5707
:5708 .    None
:5709
:5710 .SIDE EFFECTS:
:5711
:5712 .    None
:5713
:5714 .---
:5715
:5716 .SAVE.WR:
:5717
:5718 .    MOV WR[0] TO LS[SAV.WR0]      ;SAVE WR 0 IN LS LOCATION 1
:5719 .    MOV WR[1] TO LS[SAV.WR1]      ;SAVE WR 1 IN LS LOCATION 2
:5720 .    MOV WR[2] TO LS[SAV.WR2]      ;SAVE WR 2 IN LS LOCATION 3
:5721 .    MOV WR[3] TO LS[SAV.WR3]      ;SAVE WR 3 IN LS LOCATION 4
:5722 .    JMP [WAIT.SUB]                ;JUMP TO SET ATTN AND WAIT
  
```

```

U 0689, 3E02,15 :5718
U 068A, BE04,95 :5719
U 068B, 3E07,15 :5720
U 068C, 3E09,95 :5721
U 068D, 8868,74 :5722
  
```

```

:5723 .PAGE  " RESTORE WR ROUTINE"
:5724 .+++
:5725
:5726 .FUNCTIONAL DESCRIPTION:
:5727
:5728     This routine restores the 2901 working registers save in LS 1-4. The
:5729     console calls this routine before the CPU is restarted after it has
:5730     been halted by the console.
:5731     The routine simply moves the data from LS locations 1-4 to WRs 0-3
:5732     and then issues a CPU ATTN to the console. A JMP self is executed next
:5733     waiting for the console to take control.
:5734
:5735 .CALLING SEQUENCE:
:5736
:5737     The console loads the address of a pointer to this routine (a JMP in-
:5738     struction at WCS location 47) into the UPC and starts the CPU.
:5739
:5740 .INPUT PARAMETERS:
:5741
:5742     LS 1 contains the data to be restored in WR 0.
:5743     LS 2 contains the data to be restored in WR 1.
:5744     LS 3 contains the data to be restored in WR 2.
:5745     LS 4 contains the data to be restored in WR 3.
:5746
:5747 .IMPLICIT INPUTS:
:5748
:5749     None
:5750
:5751 .OUTPUT PARAMATERS:
:5752
:5753     WR 0 gets data from LS 1.
:5754     WR 1 gets data from LS 2.
:5755     WR 2 gets data from LS 3.
:5756     WR 3 gets data from LS 4.
:5757
:5758 .IMPLICIT OUTPUTS:
:5759
:5760     None
:5761
:5762 .SIDE EFFECTS:
:5763
:5764     None
:5765
:5766 .---
:5767
:5768 .RESTORE.WR:
:5769     MOV LS[SAV.WR0] TO WR[0]      :RESTORE WR 0
:5770     MOV LS[SAV.WR1] TO WR[1]      :RESTORE WR 1
:5771     MOV LS[SAV.WR2] TO WR[2]      :RESTORE WR 2
:5772     MOV LS[SAV.WR3] TO WR[3]      :RESTORE WR 3
:5773     JMP [WAIT.SUB]                :JUMP TO SET ATTN AND WAIT
  
```

U 068E, B602,15
 U 068F, 3604,95
 U 0690, B607,15
 U 0691, B609,95
 U 0692, 8868,74

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

WCS SUBROUTINES FOR 3-MAR-82
MICRO2 1L(03) 3-MAR-82 10:02:46
WCS SUBROUTINES FOR CPU USE

I 11
Fiche 1 Frame I11

Sequence 138
Rev 01.00 Page 132

```
:5774 .PAGE 'WCS SUBROUTINES FOR CPU USE'
:5775 .TOC 'ERROR ROUTINE'
:5776 .+++
:5777
:5778 .FUNCTIONAL DESCRIPTION:
:5779
:5780 .This routine is used to detect and report diagnostic errors occurring
:5781 .during WCS based micro-diagnostics. The WCS micro-code sets up the
:5782 .parameters listed below and calls this routine. The routine compares
:5783 .WR 1 (expected data) and WR 0 (received data) according to the error
:5784 .mask. Bits set in the mask indicate bits that are not checked for
:5785 .errors.
:5786 .If WR 0 and WR 1 are equal (no error) the routine executes a return+1
:5787 .to the calling point. The only exception to this is if loop on error
:5788 .is set. In that case, the error number is checked to see if that error
:5789 .occurred previously. If so, then the error loop is forced by doing a
:5790 .return. This will lock an error loop in on intermittent errors.
:5791 .If WR 0 and WR 1 are not equal (an error) then bit 8 is set in the
:5792 .CONTROL AND STATUS word, expected and received data is set up in LS for
:5793 .printout, and flags are checked in the ERROR CONTROL word. This con-
:5794 .trols whether printouts are disabled, loop on error is enabled, etc.
:5795 .After printing the error, a return+1 is made to the calling point.
:5796 .Loop on error causes a return rather than return+1 to cause an error
:5797 .loop. Also CPU ACKNOWLEDGE is toggled at the end of the routine for
:5798 .a scope sync point.
:5799
:5800 .CALLING SEQUENCE:
:5801 .
:5802 .JSR [CHECK.RESULT]
:5803
:5804 .INPUT PARAMETERS:
:5805 .
:5806 .WR[0] = Received data i.e. the actual result of the test
:5807 .WR[1] = Expected data i.e. what the result should have been
:5808
:5809 .IMPLICIT INPUTS:
:5810 .
:5811 .LS[ERROR.MASK] = Used to mask out bits that are not to be tested
:5812 .LS[ERROR.NUMBER] = Current error number
:5813 .LS[PREVIOUS.ERROR] = Error number of previous data
:5814 .LS[CONTROL.STATUS] = Control and Status word. Contains information
:5815 .written by the CPU to inform the monitor what to do.
:5816 .LS[ERROR.CONTROL] = Information such as loop-on-error, bell-on-error,
:5817 .no error report, and halt on error.
:5818
:5819 .OUTPUT PARAMATERS:
:5820 .
:5821 .NONE
:5822
:5823 .IMPLICIT OUTPUTS:
:5824 .
:5825 .LS[CONTROL.STATUS] = Control and Status with new status information
:5826 .LS[PREVIOUS.ERROR] = Last error number (modified only if error and
:5827 .loop on error is enabled)
:5828 .LS[EXPECTED.DATA] = Expected result if an error was detected
```

```

:5829 : LS[RECEIVED.DATA] = Actual result if an error was detected
:5830 :
:5831 : SIDE EFFECTS:
:5832 :
:5833 :   WR[0], WR[1], and the Q reg are modified and are not restored before
:5834 :   returning.
:5835 :   If loop-on-error is enabled and an error loop is to be executed, CPU
:5836 :   ACKNOWLEDGE will be toggled for a scope sync.
:5837 :
:5838 : ---
:5839 :
:5840 : CHECK.RESULT:
U 0693, 458A,15 :5841 BIC LS[ERROR.MASK] TO WR[0] ;MASK OFF NOT TESTED BITS (CLEAR THEM)
:5842 : ; FOR RECEIVED DATA
U 0694, C58A,95 :5843 BIC LS[ERROR.MASK] TO WR[1] ;DITTO FOR EXPECTED DATA
:5844 :
:5845 : XOR WR[0] WITH WR[1] TO Q, ;CMP RECEIVED DATA IN WR[0] WITH
U 0695, AB80,B5 :5846 DT(LONG)&SET.ALU.CC ; EXPECTED DATA IN WR[1] FOR ERROR
U 0696, 8869,F1 :5847 JMP.IF[NEQ] TO [ERROR] ;JUMP IF ERROR
:5848 :
U 0697, 3690,15 :5849 MOV LS[ERROR.CONTROL] TO WR[0] ;GET ERROR CONTROL WORD FROM LS
:5850 : BIT WR[0] WITH LS[LOE], ;SEE IF LOOP ON ERROR IS ENABLED
U 0698, 5940,35 :5851 DT(LONG)&SET.ALU.CC ; SET CONDITION CODES
U 0699, DB00,01 :5852 SKIP.IF[BIT.SET] ;SKIP IF IT IS
U 069A, DB00,16 :5853 RETURN+1 ;ELSE RETURN+1 (NO ERROR AND NO LOOP)
:5854 :
U 069B, 36A0,15 :5855 MOV LS[PREVIOUS.ERROR] TO WR[0] ;IF NO ERROR BUT LOOP ON ERROR IS
:5856 : ; ENABLED, SEE IF THERE WAS A
:5857 : ; PREVIOUS ERROR
:5858 : XOR WR[0] WITH LS[ERROR.NUMBER] TO Q,
U 069C, CD82,35 :5859 DT(LONG)&SET.ALU.CC ;IS THIS THE SAME SPOT THAT HAD AN
:5860 : ; ERROR BEFORE? (INTERMITTANT ERROR)
U 069D, 086B,91 :5861 JMP.IF[NEQ] TO [RET+1] ;JUMP IF NOT TO RETURN WITHOUT ERROR
:5862 : ; LOOPING (RETURN+1)
U 069E, 086B,C4 :5863 JMP [RET] ;ELSE CONTINUE TO LOOP (WE WILL LOOP
:5864 : ; ON ERROR EVEN IF IT GOES AWAY)
:5865 :
:5866 : ERROR:
U 069F, 3E86,15 :5867 MOV WR[0] TO LS[RECEIVED.DATA] ;PUT RESULT OF TEST IN LS FOR PRINTOUT
U 06A0, 3690,15 :5868 MOV LS[ERROR.CONTROL] TO WR[0] ;GET ERROR CONTROL BITS TO CHECK FOR
:5869 : ; LOOP COMMAND
:5870 : BIT WR[0] WITH LS[LOOP.COMMAND], ;SEE IF BIT 6 SET
U 06A1, 594C,35 :5871 DT(LONG)&SET.ALU.CC ; SET CONDITION CODES
U 06A2, 086B,C1 :5872 JMP.IF[BIT.SET] TO [RE1] ;GO LOOP IF SET (FAST LOOP)
:5873 :
U 06A3, 3E84,95 :5874 MOV WR[1] TO LS[EXPECTED.DATA] ;PUT EXPECTED DATA IN LS FOR PRINTOUT
:5875 :
U 06A4, 3680,95 :5876 MOV LS[CONTROL.STATUS] TO WR[1] ;GET CONTROL AND STATUS WORD FROM LS
U 06A5, C532,95 :5877 BIC LS[#FE7FFFFFFF] TO WR[1] ;CLEAR ALL BUT BITS 23&24 IN CONTROL
:5878 : ; AND STATUS WORD
U 06A6, C750,95 :5879 BIS LS[ERROR] TO WR[1] ;SET BIT 8 IN CONTROL AND STATUS WORD
:5880 : ; (INDICATES AN ERROR WAS DETECTED)
U 06A7, BE80,95 :5881 MOV WR[1] TO LS[CONTROL.STATUS] ;WRITE UPDATED CONTROL AND STATUS WORD
:5882 : ; BACK TO LS[40]
:5883 :

```

ZZ-ENKCB-1.0	:	ENKCB.MIC			K 11			
: ENKCB.MCR	:		MICRO2 1L(03)	ERROR ROUTINE	3-MAR-82	Fiche 1	Frame K11	Sequence 140
: ENKCB.MIC	:		ERROR ROUTINE		10:02:46	ENKCB Micro-diagnostic for DAP module	Rev 01.00	Page 134

	:5884	BIT WR[0] WITH LS[BELL],	:SEE IF BELL ON ERROR IS SET (WR 0 STILL
	:5885		:CONTAINS ERROR CONTROL WORD)
U 06A8, D944,35	:5886	DT(LONG)&SET.ALU.CC	:SET CONDITION CODES
U 06A9, 886A,D9	:5887	JMP.IF[BITS.CLR] TO [CHECK.NER]	:IF BELL ON ERROR IS NOT SET, JUMP TO
	:5888		:SEE IF ERROR REPORTING IS INHIBITED
U 06AA, 10E0,15	:5889	MISC [SET.CP.ATTN]	:ELSE SET CPU ATTN (RING MY BELL)
	:5890		
U 06AB, 086A,B4	:5891	WAIT.1: JMP [WAIT.1]	:WAIT FOR CONSOLE TO STOP ME
U 06AC, 086B,64	:5892	JMP [LOOP]	:GO TO CHECK LOOP-ON-ERROR AFTER
	:5893		:CONSOLE HAS FINISHED
	:5894		
	:5895	CHECK.NER:	
	:5896	BIT WR[0] WITH LS[NER],	:IF NO BELL SEE IF ERROR REPORT IS
	:5897		:INHIBITED
U 06AD, D942,35	:5898	DT(LONG)&SET.ALU.CC	:SET CONDITION CODES
U 06AE, 886B,21	:5899	JMP.IF[BIT.SET] TO [CHECK.HALT]	:JUMP IF ERROR REPORT IS INHIBITED TO
	:5900		:CHECK FOR HALT ON ERROR
	:5901		
U 06AF, 10E0,15	:5902	MISC [SET.CP.ATTN]	:SET CPU ATTN (REPORT ERROR)
U 06B0, 086B,04	:5903	WAIT.2: JMP [WAIT.2]	:WAIT FOR CONSOLE TO STOP ME
U 06B1, 086B,64	:5904	JMP [LOOP]	:GO TO CHECK LOOP-ON-ERROR AFTER
	:5905		:CONSOLE HAS FINISHED
	:5906		
	:5907	CHECK.HALT:	
	:5908	BIT WR[0] WITH LS[HOE],	:SEE IF HALT ON ERROR IS ENABLED
U 06B2, 5946,35	:5909	DT(LONG)&SET.ALU.CC	:SET CONDITION CODES
U 06B3, 886B,69	:5910	JMP.IF[BITS.CLR] TO [LOOP]	:JUMP IF NOT TO CHECK LOOP-ON-ERROR
	:5911		
U 06B4, 10E0,15	:5912	MISC [SET.CP.ATTN]	:ELSE SET CPU ATTN (HALT ON ERROR)
U 06B5, 086B,54	:5913	WAIT.3: JMP [WAIT.3]	:WAIT FOR CONSOLE TO STOP ME
	:5914		
U 06B6, 3690,15	:5915	LOOP: MOV LS[ERROR.CONTROL] TO WR[0]	:GET ERROR CONTROL BITS (NER, HOE ETC.)
	:5916	BIT WR[0] WITH LS[LOE],	:SEE IF LOOP-ON-ERROR
U 06B7, 5940,35	:5917	DT(LONG)&SET.ALU.CC	:SET CONDITION CODES
U 06B8, DB00,01	:5918	SKIP.IF[BIT.SET]	:SKIP IF LOOP ON ERROR IS ENABLED
	:5919		
U 06B9, DB00,16	:5920	RET+1: RETURN+1	:ELSE RETURN+1 FOR NO ERROR LOOP
	:5921		
U 06BA, 3682,15	:5922	MOV LS[ERROR.NUMBER] TO WR[0]	:GET ERROR NUMBER IF LOOPING
	:5923		
U 06BB, BEA0,15	:5924	MOV WR[0] TO LS[PREVIOUS.ERROR]	:AND SAVE IT IN PREVIOUS ERROR
	:5925		
U 06BC, 10D0,15	:5926	RET: MISC [SET.CP.ACK]	:SET CPU ACKNOWLEDGE FOR SCOPE SYNC
	:5927	MISC [CLR.CP.ATTN.AND.ACK],	:AND THEN CLEAR IT
U 06BD, 10C0,14	:5928	RETURN	:RETURN TO TEST AND LOOP ON ERROR
	:5929		

```

:5930 .PAGE
:5931
:5932
:5933
:5934 ERR.NO.TEST:
U 06BE, DB00,15 :5935 NOP ;NOP WILL CAUSE A SEQUENCE ERROR
L 06BF, B65E,15 :5936 MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR CONTROL AND
:5937 ;STATUS WORD
U 06C0, 3E80,15 :5938 MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
:5939 ;BIT 15 INDICATES START OF TEST TO
:5940 ;CONSOLE
U 06C1, 8868,74 :5941 JMP [WAIT.SUB] ;JUMP TO SET ATTN AND WAIT
:5942
:5943 .TOC
:5944 .+++
:5945 This routine loads the LS with information necessary for these tests
:5946 to run. It will then issue a CPU ATTN with the control and status word
:5947 clear to indicate that all transfers are complete.
:5948
:5949 RESULT: LS LOADED WITH CONSTANTS. MONITOR INFORMED TRANSFERS OVER.
:5950
:5951 TRANSFER.POINT:
:5952
:5953
U 06C2, 2F80,15 :5954 CLR WR[0] ;START WITH 0 IN WR[0]
U 06C3, BFFE,15 :5955 MOV WR[0] TO LS[PSL.HW] ;MAKE SURE COMPAT MODE IS CLEAR
:5956
U 06C4, 2040,15 :5957 INC WR[0] ;NOW HAVE 1 IN WR[0]
U 06C5, 23D0,15 :5958 ASHL WR[0] ;10(B)
U 06C6, 2040,15 :5959 INC WR[0] ;11(B)
U 06C7, 23D0,15 :5960 ASHL WR[0] ;110(B)
U 06C8, 2040,15 :5961 INC WR[0] ;111(B)
U 06C9, 23D0,15 :5962 ASHL WR[0] ;1110(B)
U 06CA, 23D0,15 :5963 ASHL WR[0] ;1 1100(B)
U 06CB, 23D0,15 :5964 ASHL WR[0] ;11 1000(B)
U 06CC, 23D0,15 :5965 ASHL WR[0] ;111 0000(B) NOW HAVE 70(H) IN WR 0.
:5966 ;THIS IS CORRECT START ADDRESS OF DATA
:5967 ;SECTION (INCLUDING LONGWORD COUNT,
:5968 ;DESTINATION, AND DESTINATION ADDRESS)
U 06CD, 3E0E,15 :5969 MOV WR[0] TO LS[TRANSFER.POINTER] ;LOAD LS 7 WITH POINTER TO TRANSFER
:5970 ;DATA SECTION.
:5971
U 06CE, 2F80,15 :5972 CLR WR[0] ;0 IN WR 0
U 06CF, 2040,15 :5973 INC WR[0] ;1 IN WR 0
U 06D0, 23D0,15 :5974 ASHL WR[0] ;10(B)
U 06D1, 23D0,15 :5975 ASHL WR[0] ;100(B)
U 06D2, 23D0,15 :5976 ASHL WR[0] ;1000(B)
U 06D3, 23D0,15 :5977 ASHL WR[0] ;1 0000(B)
U 06D4, 23D0,15 :5978 ASHL WR[0] ;10 0000(B)
U 06D5, 23D0,15 :5979 ASHL WR[0] ;100 0000(B)
U 06D6, 23D0,15 :5980 ASHL WR[0] ;1000 0000(B)
U 06D7, 23D0,15 :5981 ASHL WR[0] ;1 0000 0000(B)
U 06D8, 23D0,15 :5982 ASHL WR[0] ;10 0000 0000(B)
U 06D9, 23D0,15 :5983 ASHL WR[0] ;100 0000 0000(B)
U 06DA, 23D0,15 :5984 ASHL WR[0] ;1000 0000 0000(B)

```

```

U 06DB, 23D0,15 :5985      ASHL WR[0]          ;1 0000 0000 0000(B)
U 06DC, 23D0,15 :5986      ASHL WR[0]          ;10 0000 0000 0000(B)
U 06DD, 3E80,15 :5987      MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 13 IN LS CONTROL AND STATUS
:5988                                     ; WORD (BIT 13 INDICATES CONSOLE MUST
:5989                                     ; PERFORM A DATA TRANSFER)
U 06DE, 10E0,15 :5990      MISC [SET.CP.ATTN]      ;ISSUE CPU ATTN TO CONSOLE (DO FIRST LS
:5991                                     ; TRANSFER TO LOCATIONS 0 THROUGH 77(H))
:5992
U 06DF, 086D,F4 :5993      WAIT.XFER1:          ;LOOP HERE WAITING FOR CONSOLE TO RE-
:5994      JMP [WAIT.XFER1]      ; SPOND
:5995
U 06E0, 36CA,15 :5996      MOV LS[#162(H)] TO WR[0]    ;GET START ADDRESS OF SECOND HALF OF LS
:5997                                     ; TRANSFER
U 06E1, 3E0E,15 :5998      MOV WR[0] TO LS[TRANSFER.POINTER] ;LOAD START ADDRESS IN LS FOR CONSOLE
U 06E2, 365A,15 :5999      MOV LS[XFER.DATA] TO WR[0]    ;BIT 13 INDICATES TO DO DATA TRANSFER
U 06E3, 3E80,15 :6000      MOV WR[0] TO LS[CONTROL.STATUS] ;SET UP CONTROL AND STATUS WORD
U 06E4, 10E0,15 :6001      MISC [SET.CP.ATTN]      ;ISSUE CPU ATTN TO CONSOLE (DO SECOND LS
:6002                                     ; TRANSFER TO LOCATIONS 80 THROUGH F7(H))
:6003
U 06E5, 086E,54 :6004      WAIT.XFER2:          ;LOOP HERE WAITING FOR CONSOLE TO RE-
:6005      JMP [WAIT.XFER2]      ; SPOND
:6006
U 06E6, B724,15 :6007      MOV LS[HI.IPL] TO WR[0]
U 06E7, BFFE,15 :6008      MOV WR[0] TO LS[PSL.HW]      ;SET IPL TO 1F
:6009
:6010
U 06E8, 17CC,15 :6011      PORT.CONTROL [CLEAR.IDC]    ;JAM IDC TO IDLE
U 06E9, 17CC,15 :6012      PORT.CONTROL [CLEAR.IDC]
U 06EA, 17CC,15 :6013      PORT.CONTROL [CLEAR.IDC]
U 06EB, 17CC,15 :6014      PORT.CONTROL [CLEAR.IDC]    ;DONE JAMMING IDC
:6015
U 06EC, B64E,95 :6016      MOV LS[#80] TO WR[1]
U 06ED, 97A0,95 :6017      PORT.WRITE PORT.ADRS[CSR] WITH WR[1] ;SET BIT 7 OF WR 1
:6018      PORT.CONTROL [CLEAR.AUTO] ;CLEAR CRDY IN IDC
U 06EF, 97C4,15 :6019      PORT.CONTROL [RESET.BR] ;CLEAR AUTO MODE IN IDC
:6020      PORT.CONTROL [RESET.BR] ;CLEAR BR 7 IN IDC
:6021
U 06F0, E580,15 :6022      CLR LS[CONTROL.STATUS]    ;CLEAR ALL BITS OF CONTROL AND STATUS
:6023                                     ; WORD
U 06F1, 10E0,15 :6024      MISC [SET.CP.ATTN]      ;CALL CONSOLE
:6025
U 06F2, 086F,24 :6026      WAIT.XFER:          ;WAIT FOR CONSOLE TO RESPOND
:6027      JMP [WAIT.XFER]      ;USED WHEN TRANSFER IS CALLED FROM
U 06F3, 5B00,14 :6028      RETURN      ; MICRO-DIAGNOSTIC ONLY
:6029
:6030
:6031
:6032
:6033
:6034
:6035
:6036
:6037
:6038
:6039
:6040
:6041
:6042
:6043
:6044
:6045
:6046
:6047
:6048
:6049
:6050
:6051
:6052
:6053
:6054
:6055
:6056
:6057
:6058
:6059
:6060
:6061
:6062
:6063
:6064
:6065
:6066
:6067
:6068
:6069
:6070
:6071
:6072
:6073
:6074
:6075
:6076
:6077
:6078
:6079
:6080
:6081
:6082
:6083
:6084
:6085
:6086
:6087
:6088
:6089
:6090
:6091
:6092
:6093
:6094
:6095
:6096
:6097
:6098
:6099
:6100
:6101
:6102
:6103
:6104
:6105
:6106
:6107
:6108
:6109
:6110
:6111
:6112
:6113
:6114
:6115
:6116
:6117
:6118
:6119
:6120
:6121
:6122
:6123
:6124
:6125
:6126
:6127
:6128
:6129
:6130
:6131
:6132
:6133
:6134
:6135
:6136
:6137
:6138
:6139
:6140
:6141
:6142
:6143
:6144
:6145
:6146
:6147
:6148
:6149
:6150
:6151
:6152
:6153
:6154
:6155
:6156
:6157
:6158
:6159
:6160
:6161
:6162
:6163
:6164
:6165
:6166
:6167
:6168
:6169
:6170
:6171
:6172
:6173
:6174
:6175
:6176
:6177
:6178
:6179
:6180
:6181
:6182
:6183
:6184
:6185
:6186
:6187
:6188
:6189
:6190
:6191
:6192
:6193
:6194
:6195
:6196
:6197
:6198
:6199
:6200
:6201
:6202
:6203
:6204
:6205
:6206
:6207
:6208
:6209
:6210
:6211
:6212
:6213
:6214
:6215
:6216
:6217
:6218
:6219
:6220
:6221
:6222
:6223
:6224
:6225
:6226
:6227
:6228
:6229
:6230
:6231
:6232
:6233
:6234
:6235
:6236
:6237
:6238
:6239
:6240
:6241
:6242
:6243
:6244
:6245
:6246
:6247
:6248
:6249
:6250
:6251
:6252
:6253
:6254
:6255
:6256
:6257
:6258
:6259
:6260
:6261
:6262
:6263
:6264
:6265
:6266
:6267
:6268
:6269
:6270
:6271
:6272
:6273
:6274
:6275
:6276
:6277
:6278
:6279
:6280
:6281
:6282
:6283
:6284
:6285
:6286
:6287
:6288
:6289
:6290
:6291
:6292
:6293
:6294
:6295
:6296
:6297
:6298
:6299
:6300
:6301
:6302
:6303
:6304
:6305
:6306
:6307
:6308
:6309
:6310
:6311
:6312
:6313
:6314
:6315
:6316
:6317
:6318
:6319
:6320
:6321
:6322
:6323
:6324
:6325
:6326
:6327
:6328
:6329
:6330
:6331
:6332
:6333
:6334
:6335
:6336
:6337
:6338
:6339
:6340
:6341
:6342
:6343
:6344
:6345
:6346
:6347
:6348
:6349
:6350
:6351
:6352
:6353
:6354
:6355
:6356
:6357
:6358
:6359
:6360
:6361
:6362
:6363
:6364
:6365
:6366
:6367
:6368
:6369
:6370
:6371
:6372
:6373
:6374
:6375
:6376
:6377
:6378
:6379
:6380
:6381
:6382
:6383
:6384
:6385
:6386
:6387
:6388
:6389
:6390
:6391
:6392
:6393
:6394
:6395
:6396
:6397
:6398
:6399
:6400
:6401
:6402
:6403
:6404
:6405
:6406
:6407
:6408
:6409
:6410
:6411
:6412
:6413
:6414
:6415
:6416
:6417
:6418
:6419
:6420
:6421
:6422
:6423
:6424
:6425
:6426
:6427
:6428
:6429
:6430
:6431
:6432
:6433
:6434
:6435
:6436
:6437
:6438
:6439
:6440
:6441
:6442
:6443
:6444
:6445
:6446
:6447
:6448
:6449
:6450
:6451
:6452
:6453
:6454
:6455
:6456
:6457
:6458
:6459
:6460
:6461
:6462
:6463
:6464
:6465
:6466
:6467
:6468
:6469
:6470
:6471
:6472
:6473
:6474
:6475
:6476
:6477
:6478
:6479
:6480
:6481
:6482
:6483
:6484
:6485
:6486
:6487
:6488
:6489
:6490
:6491
:6492
:6493
:6494
:6495
:6496
:6497
:6498
:6499
:6500
:6501
:6502
:6503
:6504
:6505
:6506
:6507
:6508
:6509
:6510
:6511
:6512
:6513
:6514
:6515
:6516
:6517
:6518
:6519
:6520
:6521
:6522
:6523
:6524
:6525
:6526
:6527
:6528
:6529
:6530
:6531
:6532
:6533
:6534
:6535
:6536
:6537
:6538
:6539
:6540
:6541
:6542
:6543
:6544
:6545
:6546
:6547
:6548
:6549
:6550
:6551
:6552
:6553
:6554
:6555
:6556
:6557
:6558
:6559
:6560
:6561
:6562
:6563
:6564
:6565
:6566
:6567
:6568
:6569
:6570
:6571
:6572
:6573
:6574
:6575
:6576
:6577
:6578
:6579
:6580
:6581
:6582
:6583
:6584
:6585
:6586
:6587
:6588
:6589
:6590
:6591
:6592
:6593
:6594
:6595
:6596
:6597
:6598
:6599
:6600
:6601
:6602
:6603
:6604
:6605
:6606
:6607
:6608
:6609
:6610
:6611
:6612
:6613
:6614
:6615
:6616
:6617
:6618
:6619
:6620
:6621
:6622
:6623
:6624
:6625
:6626
:6627
:6628
:6629
:6630
:6631
:6632
:6633
:6634
:6635
:6636
:6637
:6638
:6639
:6640
:6641
:6642
:6643
:6644
:6645
:6646
:6647
:6648
:6649
:6650
:6651
:6652
:6653
:6654
:6655
:6656
:6657
:6658
:6659
:6660
:6661
:6662
:6663
:6664
:6665
:6666
:6667
:6668
:6669
:6670
:6671
:6672
:6673
:6674
:6675
:6676
:6677
:6678
:6679
:6680
:6681
:6682
:6683
:6684
:6685
:6686
:6687
:6688
:6689
:6690
:6691
:6692
:6693
:6694
:6695
:6696
:6697
:6698
:6699
:6700
:6701
:6702
:6703
:6704
:6705
:6706
:6707
:6708
:6709
:6710
:6711
:6712
:6713
:6714
:6715
:6716
:6717
:6718
:6719
:6720
:6721
:6722
:6723
:6724
:6725
:6726
:6727
:6728
:6729
:6730
:6731
:6732
:6733
:6734
:6735
:6736
:6737
:6738
:6739
:6740
:6741
:6742
:6743
:6744
:6745
:6746
:6747
:6748
:6749
:6750
:6751
:6752
:6753
:6754
:6755
:6756
:6757
:6758
:6759
:6760
:6761
:6762
:6763
:6764
:6765
:6766
:6767
:6768
:6769
:6770
:6771
:6772
:6773
:6774
:6775
:6776
:6777
:6778
:6779
:6780
:6781
:6782
:6783
:6784
:6785
:6786
:6787
:6788
:6789
:6790
:6791
:6792
:6793
:6794
:6795
:6796
:6797
:6798
:6799
:6800
:6801
:6802
:6803
:6804
:6805
:6806
:6807
:6808
:6809
:6810
:6811
:6812
:6813
:6814
:6815
:6816
:6817
:6818
:6819
:6820
:6821
:6822
:6823
:6824
:6825
:6826
:6827
:6828
:6829
:6830
:6831
:6832
:6833
:6834
:6835
:6836
:6837
:6838
:6839
:6840
:6841
:6842
:6843
:6844
:6845
:6846
:6847
:6848
:6849
:6850
:6851
:6852
:6853
:6854
:6855
:6856
:6857
:6858
:6859
:6860
:6861
:6862
:6863
:6864
:6865
:6866
:6867
:6868
:6869
:6870
:6871
:6872
:6873
:6874
:6875
:6876
:6877
:6878
:6879
:6880
:6881
:6882
:6883
:6884
:6885
:6886
:6887
:6888
:6889
:6890
:6891
:6892
:6893
:6894
:6895
:6896
:6897
:6898
:6899
:6900
:6901
:6902
:6903
:6904
:6905
:6906
:6907
:6908
:6909
:6910
:6911
:6912
:6913
:6914
:6915
:6916
:6917
:6918
:6919
:6920
:6921
:6922
:6923
:6924
:6925
:6926
:6927
:6928
:6929
:6930
:6931
:6932
:6933
:6934
:6935
:6936
:6937
:6938
:6939
:6940
:6941
:6942
:6943
:6944
:6945
:6946
:6947
:6948
:6949
:6950
:6951
:6952
:6953
:6954
:6955
:6956
:6957
:6958
:6959
:6960
:6961
:6962
:6963
:6964
:6965
:6966
:6967
:6968
:6969
:6970
:6971
:6972
:6973
:6974
:6975
:6976
:6977
:6978
:6979
:6980
:6981
:6982
:6983
:6984
:6985
:6986
:6987
:6988
:6989
:6990
:6991
:6992
:6993
:6994
:6995
:6996
:6997
:6998
:6999
:7000
:7001
:7002
:7003
:7004
:7005
:7006
:7007
:7008
:7009
:7010
:7011
:7012
:7013
:7014
:7015
:7016
:7017
:7018
:7019
:7020
:7021
:7022
:7023
:7024
:7025
:7026
:7027
:7028
:7029
:7030
:7031
:7032
:7033
:7034
:7035
:7036
:7037
:7038
:7039
:7040
:7041
:7042
:7043
:7044
:7045
:7046
:7047
:7048
:7049
:7050
:7051
:7052
:7053
:7054
:7055
:7056
:7057
:7058
:7059
:7060
:7061
:7062
:7063
:7064
:7065
:7066
:7067
:7068
:7069
:7070
:7071
:7072
:7073
:7074
:7075
:7076
:7077
:7078
:7079
:7080
:7081
:7082
:7083
:7084
:7085
:7086
:7087
:7088
:7089
:7090
:7091
:7092
:7093
:7094
:7095
:7096
:7097
:7098
:7099
:7100
:7101
:7102
:7103
:7104
:7105
:7106
:7107
:7108
:7109
:7110
:7111
:7112
:7113
:7114
:7115
:7116
:7117
:7118
:7119
:7120
:7121
:7122
:7123
:7124
:7125
:7126
:7127
:7128
:7129
:7130
:7131
:7132
:7133
:7134
:7135
:7136
:7137
:7138
:7139
:7140
:7141
:7142
:7143
:7144
:7145
:7146
:7147
:7148
:7149
:7150
:7151
:7152
:7153
:7154
:7155
:7156
:7157
:7158
:7159
:7160
:7161
:7162
:7163
:7164
:7165
:7166
:7167
:7168
:7169
:7170
:7171
:7172
:7173
:7174
:7175
:7176
:7177
:7178
:7179
:7180
:7181
:7182
:7183
:7184
:7185
:7186
:7187
:7188
:7189
:7190
:7191
:7192
:7193
:7194
:7195
:7196
:7197
:7198
:7199
:7200
:7201
:7202
:7203
:7204
:7205
:7206
:7207
:7208
:7209
:7210
:7211
:7212
:7213
:7214
:7215
:7216
:7217
:7218
:7219
:7220
:7221
:7222
:7223
:7224
:7225
:7226
:7227
:7228
:7229
:7230
:7231
:7232
:7233
:7234
:7235
:7236
:7237
:7238
:7239
:7240
:7241
:7242
:7243
:7244
:7245
:7246
:7247
:7248
:7249
:7250
:7251
:7252
:7253
:7254
:7255
:7256
:7257
:7258
:7259
:7260
:7261
:7262
:7263
:7264
:7265
:7266
:7267
:7268
:7269
:7270
:7271
:7272
:7273
:7274
:7275
:7276
:7277
:7278
:7279
:7280
:7281
:7282
:7283
:7284
:7285
:7286
:7287
:7288
:7289
:7290
:7291
:7292
:7293
:7294
:7295
:7296
:7297
:7298
:7299
:7300
:7301
:7302
:7303
:7304
:7305
:7306
:7307
:7308
:7309
:7310
:7311
:7312
:7313
:7314
:7315
:7316
:7317
:7318
:7319
:7320
:7321
:7322
:7323
:7324
:7325
:7326
:7327
:7328
:7329
:7330
:7331
:7332
:7333
:7334
:7335
:7336
:7337
:7338
:7339
:7340
:7341
:7342
:7343
:7344
:7345
:7346
:7347
:7348
:7349
:7350
:7351
:7352
:7353
:7354
:7355
:7356
:7357
:7358
:7359
:7360
:7361
:7362
:7363
:7364
:7365
:7366
:7367
:7368
:7369
:7370
:7371
:7372
:7373
:7374
:7375
:7376
:7377
:7378
:7379
:7380
:7381
:7382
:7383
:7384
:7385
:7386
:7387
:7388
:7389
:7390
:7391
:7392
:7393
:7394
:7395
:7396
:7397
:7398
:7399
:7400
:7401
:7402
:7403
:7404
:7405
:7406
:7407
:7408
:7409
:7410
:7411
:7412
:7413
:7414
:7415
:7416
:7417
:7418
:7419
:7420
:7421
:7422
:7423
:7424
:7425
:7426
:7427
:7428
:7429
:7430
:7431
:7432
:7433
:7434
:7435
:7436
:7437
:7438
:7439
:7440
:7441
:7442
:7443
:7444
:7445
:7446
:7447
:7448
:7449
:7450
:7451
:7452
:7453
:7454
:7455
:7456
:7457
:7458
:7459
:7460
:7461
:7462
:7463
:7464
:7465
:7466
:7467
:7468
:7469
:7470
:7471
:7472
:7473
:7474
:7475
:7476
:7477
:7478
:7479
:7480
:7481
:7482
:7483
:7484
:7485
:7486
:7487
:7488
:7489
:7490
:7491
:7492
:7493
:7494
:7495
:7496
:7497
:7498
:7499
:7500
:7501
:7502
:7503
:7504
:7505
:7506
:7507
:7508
:7509
:7510
:7511
:7512
:7513
:7514
:7515
:7516
:7517
:7518
:7519
:7520
:7521
:7522
:7523
:7524
:7525
:7526
:7527
:7528
:7529
:7530
:7531
:7532
:7533
:7534
:7535
:7536
:7537
:7538
:7539
:7540
:7541
:7542
:7543
:7544
:7545
:7546
:7547
:7548
:7549
:7550
:7551
:7552
:7553
:7554
:7555
:7556
:7557
:7558
:7559
:7560
:7561
:7562
:7563
:7564
:7565
:7566
:7567
:7568
:7569
:7570
:7571
:7572
:7573
:7574
:7575
:7576
:7577
:7578
:7579
:7580
:7581
:7582
:7583
:7584
:7585
:7586
:7587
:7588
:7589
:7590
:7591
:7592
:7593
:7594
:7595
:7596
:7597
:7598
:7599
:7600
:7601
:7602
:7603
:7604
:7605
:7606
:7607
:7608
:7609
:7610
:7611
:7612
:7613
:7614
:7615
:7616
:7617
:7618
:7619
:7620
:7621
:7622
:7623
:7624
:7625
:7626
:7627
:7628
:7629
:7630
:7631
:7632
:7633
:7634
:7635
:7636
:7637
:7638
:7639
:7640
:7641
:7642
:7643
:7644
:7645
:7646
:7647
:7648
:7649
:7650
:7651
:7652
:7653
:7654
:7655
:7656
:7657
:7658
:7659
:7660
:7661
:7662
:7663
:7664
:7665
:7666
:7667
:7668
:7669
:7670
:7671
:7672
:7673
:7674
:7675
:7676
:7677
:7678
:7679
:7680
:7681
:7682
:7683
:7684
:7685
:7686
:7687
:7688
:7689
:7690
:7691
:7692
:7693
:7694
:7695
:7696
:7697
:7698
:7699
:7700
:7701
:7702
:7703
:7704
:7705
:7706
:7707
:7708
:7709
:7710
:7711
:7712
:7713
:7714
:7715
:7716
:7717
:7718
:7719
:7720
:7721
:7722
:7723
:7724
:7725
:7726
:7727
:7728
:7729
:7730
:7731
:7732
:7733
:7734
:7735
:7736
:7737
:7738
:7739
:7740
:7741
:7742
:7743
:7744
:7745
:7746
:7747
:7748
:7749
:7750
:7751
:7752
:7753
:7754
:7755
:7756
:7757
:7758
:7759
:7760
:7761
:7762
:7763
:7764
:7765
:7766
:7767
:7768
:7769
:7770
:7771
:7772
:7773
:7774
:7775
:7776
:7777
:7778
:7779
:7780
:7781
:7782
:7783
:7784
:7785
:7786
:7787
:7788
:7789
:7790
:7791
:7792
:7793
:7794
:7795
:7796
:7797
:7798
:7799
:7800
:7801
:7802
:7803
:7804
:7805
:7806
:7807
:7808
:7809
:7810
:7811
:7812
:7813
:7814
:7815
:7816
:7817
:7818
:7819
:7820
:7821
:7822
:7823
:7824
:7825
:7826
:7827
:7828
:7829
:7830
:7831
:7832
:7833
:7834
:7835
:7836
:7837
:7838
:7839
:7840
:7841
:7842
:7843
:7844
:7845
:7846
:7847
:7848
:7849
:7850
:7851
:7852
:7853
:7854
:7855
:7856
:7857
:7
```


ZZ-ENKCB-1.0 ; ENKCB.MIC START TRANSFER ROUT N 11
: ENKCB.MCR MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Sequence 143
: ENKCB.MIC START TRANSFER ROUTINE Rev 01.00 Page 137

:6029 .PAGE
:6030 .REGION/700,3FFF
:6031

```

:6032 .Page " General Use Subroutines"
:6033
:6034 ;This subroutine obtains the error number and increments it by one.
:6035 INC.EN:
U 0700, 3682,15 :6036 MOV LS[ERROR.NUMBER] TO WR[0] ;GET ERROR NUMBER
U 0701, 2040,15 :6037 INC WR[0] ; AND INCREMENT
L 0702, BE82,15 :6038 MOV WR[0] TO LS[ERROR.NUMBER] ; IT.
U 0703, 5B00,14 :6039 RETURN
:6040
:6041 ;This subroutine obtains the error number and decrements it by one.
:6042 DEC.EN:
U 0704, 3682,15 :6043 MOV LS[ERROR.NUMBER] TO WR[0] ;GET ERROR NUMBER
U 0705, 2100,15 :6044 DEC WR[0] ; AND DECREMENT
U 0706, BE82,15 :6045 MOV WR[0] TO LS[ERROR.NUMBER] ; IT.
U 0707, 5B00,14 :6046 RETURN
:6047
:6048 ;This subroutine asserts the N/A flag for expected and received data fields.
:6049 ASSERT.NA:
U 0708, B66E,15 :6050 MOV LS[BIT23] TO WR[0]
U 0709, 6D80,15 :6051 BIS WR[0] TO LS[CONTROL.STATUS] ;CSR = BIT23
U 070A, 5B00,14 :6052 RETURN
:6053
:6054 ;This subroutine deasserts the N/A flag for expected and received data fields.
:6055 DEASSERT.NA:
U 070B, B680,15 :6056 MOV LS[CONTROL.STATUS] TO WR[0]
U 070C, 456E,15 :6057 BIC LS[BIT23] TO WR[0]
U 070D, 3E80,15 :6058 MOV WR[0] TO LS[CONTROL.STATUS]
U 070E, 5B00,14 :6059 RETURN
:6060
:6061 ;This subroutine asserts the flag for "other" data field.
:6062 ASSERT.OTHER:
U 070F, B670,15 :6063 MOV LS[BIT24] TO WR[0]
U 0710, 6D80,15 :6064 BIS WR[0] TO LS[CONTROL.STATUS]
U 0711, 5B00,14 :6065 RETURN
:6066
:6067
:6068 ;This subroutine deasserts the flag for "other" data field.
:6069 DEASSERT.OTHER:
U 0712, B680,15 :6070 MOV LS[CONTROL.STATUS] TO WR[0]
U 0713, 4570,15 :6071 BIC LS[BIT24] TO WR[0]
U 0714, 3E80,15 :6072 MOV WR[0] TO LS[CONTROL.STATUS]
U 0715, 5B00,14 :6073 RETURN
:6074
:6075
:6076 718: ;THIS LOCATION IS RESERVED FOR USE BY TEST #F.
:6077 RETURN.718:
U 0718, 5B00,14 :6078 RETURN
:6079
  
```

:6080 Page "TEST 1 - Stack RAM Data Test (M8390 CPU Module)"

:6081 ++

:6082
:6083 Test Description:

:6084
:6085 This test checks each of the 16 locations of the stack RAMS for
:6086 data integrity. One location of the stack was tested in the JSR
:6087 and RETURN test and has been used for all subroutine calls up to
:6088 this point. The other stack RAM locations can be checked by ex-
:6089 ecuting a POP to discard the top entry before issuing a JSR. Each
:6090 time the stack is popped, the stack address used to save the return
:6091 uPC will be different. When the test has executed 16 times, all
:6092 stack locations will have been tested for data integrity. The POP
:6093 function has not been tested yet, but if it doesn't work, it will
:6094 mean the same RAM location will be used 16 times. The POP instruction
:6095 will be tested at the end of this test. This test will loop 16
:6096 times, executing one POP before each time through the loop to change
:6097 the stack RAM address. The data path is as follows: After an
:6098 initial POP has been executed, A JSR will be performed with the
:6099 address of the JSR instruction set up so as to shift a 0 across a
:6100 field of ones in the stack RAM. A RETURN will be performed at the
:6101 destination address to POP the last entry off the stack and use it as
:6102 the return PC. This will be repeated until a 0 has been shifted
:6103 across uPC bits 0-13. The test is then repeated with a new stack RAM
:6104 address. The POP instruction will be tested last to assure that at
:6105 least 2 unique stack RAM locations can be accessed and that the POP
:6106 instruction itself is working. The POP instruction will be tested by
:6107 performing two JSR's, doing a POP, and executing a RETURN. Then a
:6108 check is performed to see that the return went to the first JSR
:6109 location + 1 correctly.

:6110
:6111 Assumptions:

:6112
:6113 It is assumed that the previous JSR and RETURN test has been success-
:6114 fully run.

:6115
:6116 Since this test is checking for a return to a specific address, not
:6117 data, the expected and received data in the error printout should be
:6118 ignored.

:6119
:6120 NOTE: A stack failure in the initial phase of each loop could send
:6121 u-code into never-never land. Single stepping is the only logical
:6122 way to trace this type of problem.

:6123
:6124 Test Steps:

- :6125
:6126 1) Set up error mask, error number, and module number in LS (for
:6127 error printouts) and clear previous error number in LS.
:6128 2) Execute a POP to decrement the stack pointer address.
:6129 3) Execute a JSR at a u-address that will push 1's on the stack
:6130 except for 1 bit.
:6131 4) Execute a RETURN at the destination address. A failure at this
:6132 point will almost certainly cause the u-code to get lost!
:6133 5) At the return address, jump to a location that will shift a 0
:6134 across the stack output and execute a JSR.

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.P.C

D 12
TEST 1 - Stack RAM 3-MAR-82
MICRO2 1L(03) 3-MAR-82 10:02:46
TEST 1 - Stack RAM Data Test (M8390 CPU Module)

Fiche 1 Frame D12

Sequence 146
Rev 01.00 Page 140

:6135 : 6) Execute a RETURN at the destination address. Report error if the
:6136 : return goes to the old address.
:6137 : 7) Repeat steps 5 and 6 until a zero has shifted across address bits
:6138 : 0-13.
:6139 : 8) Repeat steps 2 through 7 until all 16 locations of the stack RAM
:6140 : have been tested.
:6141 : 9) Execute two JSR's and then execute a POP instruction.
:6142 : 10) Execute a RETURN and check that a return to the first JSR location
:6143 : + 1 occurs.
:6144 :
:6145 :
:6146 :
:6147 :
:6148 :
:6149 :
:6150 :
:6151 :
:6152 :
:6153 :
:6154 :
:6155 :
:6156 :
:6157 :
:6158 :
:6159 :
:6160 :
:6161 :
:6162 :
:6163 :
:6164 :
:6165 :
:6166 :
:6167 :
:6168 :
:6169 :
:6170 :
:6171 :
:6172 :
:6173 :
:6174 :
:6175 :
:6176 :
:6177 :
:6178 :
:6179 :
:6180 :
:6181 :
:6182 :
:6183 :
:6184 :
:6185 :
:6186 :
:6187 :
:6188 :
:6189 :

Errors:

Error 1 - Stack RAM failure on return; bit 13.
Error 2 - Stack RAM failure on return; bit 12.
Error 3 - Stack RAM failure on return; bit 11.
Error 4 - Stack RAM failure on return; bit 10.
Error 5 - Stack RAM failure on return; bit 9.
Error 6 - Stack RAM failure on return; bit 8.
Error 7 - Stack RAM failure on return; bit 7.
Error 8 - Stack RAM failure on return; bit 6.
Error 9 - Stack RAM failure on return; bit 5.
Error A - Stack RAM failure on return; bit 4.
Error B - Stack RAM failure on return; bit 3.
Error C - Stack RAM failure on return; bit 2.
Error D - Stack RAM failure on return; bit 1.
Error E - Stack RAM failure on return; bit 0.
Error F - POP failed to decrement address of stack pointer.

Debug:

NOTE: In the event of a failure, the specific stack address that failed can not be ascertained by the u-code. Halting on error and scoping the current address is the only means to determine this. However the actual address doesn't matter, as only the data bits coming out of the stack RAM are being tested here.

Error 1 through E - The most likely suspect is the stack RAM itself as the JSR and RETURN functions themselves have already been tested. The failing bit in the stack RAM can be determined by the error number and the correct stack RAM can then be ascertained.

Error F - This error indicates a problem with the SEQ.CTL PAL or the STACK POINTER PAL. Check the output of the SEQ.CTL PAL during the POP instruction for a low on ENABLE SP L. If this is incorrect, check the inputs to the PAL for a 12(H) on CSR 4-0. The PAL is bad if these inputs are OK. If ENABLE SP L is OK, check it for a low at the input to the STACK POINTER PAL. Also check CSR 03 H for a high. If these inputs are OK, the STACK POINTER PAL is probably bad. However the address inputs to the stack RAM's should also be checked to assure that an etch break has not occurred (look for a float on pins 3 through 6 of each stack RAM).

U 0719, B65E,15

T.1:

MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT15 IN WRO FOR CONTROL AND STATUS WORD

```

U 071A, 3E80,15 :6190      MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT15 IN CNTRL AND STATUS WORD
:6191
U 071B, 10E0,15 :6192      MISC [SET.CP.ATTN] ;BIT15 INDICATES START OF TEST TO CONSOLE
:6193      WAIT.T1.0: ;ISSUE CPU ATTN TO CONSOLE
U 071C, 8871,14 :6194      JMP [WAIT.T1.0] ;LOOP HERE WAITING FOR CONSOLE TO RESPOND
U 071D, E58A,15 :6195      CLR LS[ERROR.MASK] ;CLEAR ERROR MASK.
U 071E, 65A0,15 :6196      CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
U 071F, 2F80,15 :6197      CLR WR[0]
U 0720, 2040,15 :6198      INC WR[0] ;SET WRO TO 1
U 0721, BE82,15 :6199      MOV WR[0] TO LS[ERROR.NUMBER] ;SET ERROR NUMBER TO FIRST ERROR
U 0722, A3C0,15 :6200      RCL WR[0] ;SET WRO TO 2
U 0723, 3E8C,15 :6201      MOV WR[0] TO LS[MODULE.NUM] ;SET UP MODULE CODE FOR CPU MODULE
U 0724, B66E,15 :6202      MOV LS[BIT23] TO WR[0]
U 0725, 3E80,15 :6203      MOV WR[0] TO LS[CONTROL.STATUS] ;BIT23 = EXPECTED AND RECEIVED DATA N/A
U 0726, 2F85,15 :6204      CLR WR[2] ;SET BIGLOOP COUNTER TO 0
:6205      BIGLOOP.T1:
U 0727, 2045,15 :6206      INC WR[2] ;INCREMENT BIG LOOP COUNTER
U 0728, 5B00,12 :6207      SKIP.IF[POP.USTACK] ;NOTE: THIS DOES NOT SKIP; THIS IS THE
:6208      ; POP INSTRUCTION BELIEVE IT OR NOT!
:6209      LOOP.T1.1:
U 0729, B640,15 :6210      MOV LS[#1] TO WR[0]
U 072A, BE82,15 :6211      MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR NUMBER = 1
U 072B, 89FF,E4 :6212      JMP [L.1FFE]
:6213      1FFE:
:6214      L.1FFE:
:6215      ;The following will cause a validity failure:
U 1FFE, 8950,AC :6216      JSR [SUB.T1] ;PUSH RET ADD OF 1FFF ONTO STACK
U 1FFF, 8870,0C :6217      JSR [INC.EN] ;INCREMENT ERROR # TO 2
U 2000, 8AFF,E4 :6218      JMP [T1.2]
:6219      2FFE:
:6220      T1.2:
:6221      ;The following will cause a validity failure:
U 2FFE, 8950,AC :6222      JSR [SUB.T1] ;PUSH RET ADD OF 2FFF ONTO STACK
U 2FFF, 8870,0C :6223      JSR [INC.EN] ;INCREMENT ERROR # TO 3
U 3000, 8B7F,E4 :6224      JMP [T1.3]
:6225      37FE:
:6226      T1.3:
:6227      ;The following will cause a validity failure:
U 37FE, 8950,AC :6228      JSR [SUB.T1] ;PUSH RET ADD OF 37FF ONTO STACK
U 37FF, 8870,0C :6229      JSR [INC.EN] ;INCREMENT ERROR # TO 4
U 3800, 8B8F,E4 :6230      JMP [T1.4]
:6231      3BFE:
:6232      T1.4:
U 3BFE, 8950,AC :6233      JSR [SUB.T1] ;PUSH RET ADD OF 3BFF ONTO STACK
U 3BFF, 8870,0C :6234      JSR [INC.EN] ;INCREMENT ERROR # TO 5
U 3C00, 8BDF,E4 :6235      JMP [T1.5]
:6236      3DFE:
:6237      T1.5:
U 3DFE, 8950,AC :6238      JSR [SUB.T1] ;PUSH RET ADD OF 3DFF ONTO STACK
U 3DFF, 8870,0C :6239      JSR [INC.EN] ;INCREMENT ERROR # TO 6
U 3E00, 8BEF,E4 :6240      JMP [T1.6]
:6241      3EFE:
  
```

```

:6242 T1.6:
U 3EFE, 8950.AC :6243 JSR [SUB.T1] ;PUSH RET ADD OF 3EFF ONTO STACK
U 3EFF, 8870.0C :6244 JSR [INC.EN] ;INCREMENT ERROR # TO 7
U 3F00, 8BF7.E4 :6245 JMP [T1.7]
:6246 3F7E:
:6247 T1.7:
U 3F7E, 8950.AC :6248 JSR [SUB.T1] ;PUSH RET ADD OF 3F7F ONTO STACK
U 3F7F, 8870.0C :6249 JSR [INC.EN] ;INCREMENT ERROR # TO 8
U 3F80, 8BFB.E4 :6250 JMP [T1.8]
:6251 3FBE:
:6252 T1.8:
U 3FBE, 8950.AC :6253 JSR [SUB.T1] ;PUSH RET ADD OF 3FBF ONTO STACK
U 3FBF, 8870.0C :6254 JSR [INC.EN] ;INCREMENT ERROR # TO 9
U 3FC0, 8BFD.E4 :6255 JMP [T1.9]
:6256 3FDE:
:6257 T1.9:
U 3FDE, 8950.AC :6258 JSR [SUB.T1] ;PUSH RET ADD OF 3FDF ONTO STACK
U 3FDF, 8870.0C :6259 JSR [INC.EN] ;INCREMENT ERROR # TO A
U 3FEO, 8BFE.E4 :6260 JMP [T1.A]
:6261 3FEE:
:6262 T1.A:
U 3FEE, 8950.AC :6263 JSR [SUB.T1] ;PUSH RET ADD OF 3FEF ONTO STACK
U 3FEF, 8870.0C :6264 JSR [INC.EN] ;INCREMENT ERROR # TO B
U 3FF0, 8BFF.64 :6265 JMP [T1.B]
:6266 3FF6:
:6267 T1.B:
U 3FF6, 8950.AC :6268 JSR [SUB.T1] ;PUSH RET ADD OF 3FF7 ONTO STACK
U 3FF7, 8870.0C :6269 JSR [INC.EN] ;INCREMENT ERROR # TO C
U 3FF8, 8BFF.A4 :6270 JMP [T1.C]
:6271 3FFA:
:6272 T1.C:
U 3FFA, 8950.AC :6273 JSR [SUB.T1] ;PUSH RET ADD OF 3FFB ONTO STACK
U 3FFB, 8870.0C :6274 JSR [INC.EN] ;INCREMENT ERROR # TO D
U 3FFC, 8B5F.C4 :6275 JMP [T1.D]
:6276 35FC:
:6277 T1.D:
U 35FC, 8950.AC :6278 JSR [SUB.T1] ;PUSH RET ADD OF 3FFD ONTO STACK
U 35FD, 8870.0C :6279 JSR [INC.EN] ;INCREMENT ERROR # TO E
U 35FE, 0B6F.D4 :6280 JMP [T1.E]
:6281 36FD:
:6282 T1.E:
U 36FD, 8950.AC :6283 JSR [SUB.T1] ;PUSH RET ADD OF 36FE ONTO STACK
U 36FE, 0950.54 :6284 JMP [T1.EA]
:6285 3FFF:
:6286 T1.ERR.1:
U 3FFF, 0950.04 :6287 JMP [T1.ERR.1]
:6288 1500:
:6289 T1.ERR.1:
U 1500, B69E.15 :6289 MOV LS[ONES] TO WR[0] ;ERROR IF HERE SO SET UP DELIBERATELY
U 1501, 2F82.95 :6290 CLR WR[1] ;ERRONEOUS DATA TO FLAG THE ERROR
U 1502, 0869.3C :6291 JSR [CHECK.RESULT]
U 1503, 8872.94 :6292 JMP [LOOP.T1.1] ;LOOP ON ERROR IF ENABLED
U 1504, 0950.54 :6293 JMP [T1.EA]
:6294 T1.EA:
:6295
U 1505, 4D49.35 :6296 XOR LS[10] WITH WR[2] TO 0, ;IS BIG LOOP COUNTER = 16 ?
DT(LONG)&SET.ALU.CC

```

G 12

ZZ-ENKCB-1.0 : ENKCB.MIC TEST 1 - Stack RAM 3-MAR-82 Fiche 1 Frame G12 Sequence 149
 : ENKCB.MCR MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Rev 01.00 Page 143
 : ENKCB.MIC TEST 1 - Stack RAM Data Test (M8390 CPU Module)

```

U 1506, 0872.71 :6297      JMP.[IF[NEQ] TO [BIGLOOP.T1]      ;LOOP BACK IF NOT
U 1507, 8870.0C :6298      JSR [INC.EN]                ;INCREMENT ERROR NUMBER TO F
                                :6299
U 1508, 0950.BC :6300      LOOP.T1.F:                ;JUMP TO SUBROUTINE A
U 1509, 8952.04 :6301      JSR [SUB.T1.A]                ;JUMP OVER SUBROUTINES
                                :6302
                                SUB.T1:                ;DUMMY SUBROUTINE
U 150A, 5B00.14 :6303      RETURN
                                :6304
                                SUB.T1.A:                ;CALL SUBROUTINE B
U 150B, 8951.1C :6305      JSR [SUB.T1.B]                ;ERROR IF HERE SO SET UP DELIBERATELY
U 150C, 369E.95 :6306      MOV LS[ONES] TO WR[1]          ; ERRONEOUS DATA TO FLAG THE ERROR
U 150D, 2F80.15 :6307      CLR WR[0]
U 150E, 0869.3C :6308      JSR [CHECK.RESULT]
U 150F, 8950.84 :6309      JMP [LOOP.T1.F]                ;LOOP ON ERROR IF ENABLED
U 1510, 8952.04 :6310      JMP [END.T1]
                                :6311
                                SUB.T1.B:
U 1511, 5B00.12 :6312      SKIP.[IF[POP.USTACK]
                                :6313
U 1512, 5B00.14 :6314      RETURN
                                :6314
  
```

; NOTE: THIS DOES NOT SKIP; THIS IS THE
; POP INSTRUCTION BELIEVE IT OR NOT!

	:6315	.PAGE	
	:6316	818:	;THIS LOCATION IS RESERVED FOR USE BY TEST #F.
	:6317	RETURN.818:	
U 0818, 5B00,14	:6318	RETURN	
	:6319	C18:	;THIS LOCATION IS RESERVED FOR USE BY TEST #F.
	:6320	RETURN.C18:	
L 0819, 5B00,14	:6321	RETURN	
	:6322	1018:	;THIS LOCATION IS RESERVED FOR USE BY TEST #F.
	:6323	RETURN.1018:	
U 1018, 5B00,14	:6324	RETURN	
	:6325	1418:	;THIS LOCATION IS RESERVED FOR USE BY TEST #F.
	:6326	RETURN.1418:	
U 1418, 5B00,14	:6327	RETURN	
	:6328	1A18:	;THIS LOCATION IS RESERVED FOR USE BY TEST #F.
	:6329	RETURN.1A18:	
U 1A18, 5B00,14	:6330	RETURN	
	:6331	1C18:	;THIS LOCATION IS RESERVED FOR USE BY TEST #F.
	:6332	RETURN.1C18:	
U 1C18, 5B00,14	:6333	RETURN	
	:6334	2018:	;THIS LOCATION IS RESERVED FOR USE BY TEST #F.
	:6335	RETURN.2018:	
U 2018, 5B00,14	:6336	RETURN	
	:6337	2418:	;THIS LOCATION IS RESERVED FOR USE BY TEST #F.
	:6338	RETURN.2418:	
U 2418, 5B00,14	:6339	RETURN	
	:6340	2818:	;THIS LOCATION IS RESERVED FOR USE BY TEST #F.
	:6341	RETURN.2818:	
U 2818, 5B00,14	:6342	RETURN	
	:6343	2C18:	;THIS LOCATION IS RESERVED FOR USE BY TEST #F.
	:6344	RETURN.2C18:	
U 2C18, 5B00,14	:6345	RETURN	
	:6346	3018:	;THIS LOCATION IS RESERVED FOR USE BY TEST #F.
	:6347	RETURN.3018:	
U 3018, 5B00,14	:6348	RETURN	
	:6349	3418:	;THIS LOCATION IS RESERVED FOR USE BY TEST #F.
	:6350	RETURN.3418:	
U 3418, 5B00,14	:6351	RETURN	
	:6352	3818:	;THIS LOCATION IS RESERVED FOR USE BY TEST #F.
	:6353	RETURN.3818:	
U 3818, 5B00,14	:6354	RETURN	
	:6355	3C18:	;THIS LOCATION IS RESERVED FOR USE BY TEST #F.
	:6356	RETURN.3C18:	
U 3C18, 5B00,14	:6357	RETURN	
	:6358	1520:	
	:6359	END.T1:	

:6360 Page "TEST 2 - Nested Subroutines Test (M8390 CPU Module)"

:6361 ++

:6362 Test Description:

:6363 This test checks the STACK POINTER PAL and the stack RAMS to assure
:6364 that 16 unique locations can be accessed for subroutine nesting.
:6365 A series of 16 JSR's will be performed followed by 16 RETURNS.
:6366 Each time a RETURN is executed, WR 0 will be incremented and
:6367 checked at the return address for the correct value. This will
:6368 prevent an endless loop if the RAM addresses are overwritten.
:6369 After the stack addressing has been checked, the last test checks
:6370 that the RETURN + 1 instruction works also.

:6371 Assumptions:

:6372 It is assumed that the previous stack RAM test has run successfully.

:6373 Test Steps:

- :6374 1) Set up error mask, error number, and module number in LS (for
:6375 error printouts) and clear previous error number in LS.
:6376 2) Clear WR 0 and then execute 16 JSR's.
:6377 3) At the last JSR destination, increment WR 0 and perform a RETURN.
:6378 4) Check WR 0 for the correct value, increment WR 0 again, and execute
:6379 a RETURN.
:6380 5) Repeat step 4 until WR 0 = 16 decimal.
:6381 6) Do 2 JSR's and 2 RETURN + 1's to make sure RETURN + 1 is post
:6382 incrementing the stack address also.

:6383 Errors:

- :6384 Error 1 - Stack RAM or STACK POINTER PAL failed.
:6385 Error 2 - RETURN + 1 failed to decrement stack pointer correctly.

:6386 Debug:

:6387 Error 1 - The expected data printout of the error report tells which
:6388 location should have been returned to (see listing). The most likely
:6389 cause of an error is that the stack RAM locations were overwritten
:6390 during the 16 JSR instructions. If an address line never changes,
:6391 then the same location will be written twice. One possibility is that
:6392 the SEQ.CTL PAL is not asserting ENABLE SP L on the return instruction.
:6393 Check ENABLE SP L at the STACK POINTER PAL. It should be low during
:6394 the RETURN instruction. If not the SEQ.CTL PAL is bad. A more
:6395 likely possibility is that the STACK POINTER PAL is faulty. The
:6396 easiest way to check this PAL is to single step through the JSR in-
:6397 structions while scopeing the address lines. The same procedure can
:6398 be performed during the RETURN instructions if the JSR's work OK.
:6399 A third possibility is that an address line is not making it to the
:6400 stack RAMS or that the stack RAM itself is failing internally.

:6401 Error 2 - If this is the first error, the SEQ.CTL PAL is the most
:6402 likely suspect. This error indicates that RETURN does the post
:6403 increment after the pop correctly, but RETURN + 1 does not.

```

:6415
:6416
:6417
:6418
:6419
:6420
:6421
:6422
:6423
:6424
:6425
:6426
:6427
:6428
:6429
:6430
:6431
:6432
:6433
:6434
:6435
:6436
:6437
:6438
:6439
:6440
:6441
:6442
:6443
:6444
:6445
:6446
:6447
:6448
:6449
:6450
:6451
:6452
:6453
:6454
:6455
:6456
:6457
:6458
:6459
:6460
:6461
:6462
:6463
:6464
:6465
:6466
:6467
:6468
:6469
    
```

```

T.2:
    MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT15 IN WRO FOR CONTROL AND STATUS WORD
    MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT15 IN CNTRL AND STATUS WORD
    MISC [SET.CP.ATTN] ;BIT15 INDICATES START OF TEST TO CONSOLE
    ;ISSUE CPU ATTN TO CONSOLE
WAIT.T2.0:
    JMP [WAIT.T2.0] ;LOOP HERE WAITING FOR CONSOLE TO RESPOND
    CLR LS[ERROR.MASK] ;CLEAR ERROR MASK.
    CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
    INC WR[0] ;SET WRO TO 1
    MOV WR[0] TO LS[ERROR.NUMBER] ;SET ERROR NUMBER TO FIRST ERROR
    ROL WR[0] ;SET WRO TO 2
    MOV WR[0] TO LS[MODULE.NUM] ;SET UP MODULE CODE FOR CPU MODULE
    MOV LS[BIT23] TO WR[0]
    MOV WR[0] TO LS[CONTROL.STATUS] ;BIT23 = EXPECTED AND RECEIVED DATA N/A
LOOP.T2.1:
    CLR WR[0] ;SET WR TO 0
    JSR [T2.SUB.1] ;ISSUE 1ST OF 16 NESTED CALLS
    MOV LS[#10] TO WR[1] ;EXPECTED DATA = 17
    INC WR[1]
    JSR [CHECK.RESULT] ;SEE IF WRO WAS INCREMENTED 16 TIMES
    JMP [LOOP.T2.1] ;LOOP ON ERROR IF ENABLED
    JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 2
LOOP.T2.2:
    CLR WR[0]
    JSR [T2.SUB.17] ;CALL FIRST OF TWO NESTED CALLS
    CLR WR[0] ;ERROR IF HERE
    MOV LS[#2] TO WR[1] ;EXPECTED DATA = 2
    JSR [CHECK.RESULT] ;CHECK TO SEE BOTH RETURN + 1 'S WORKED
    JMP [LOOP.T2.2] ;LOOP ON ERROR IF ENABLED
    JMP [END.T2] ;ALL DONE - SKIP OVER SUBROUTINES
T2.SUB.17:
    JSR [T2.SUB.18] ;CALL SECOND OF 2 NESTED CALLS
    CLR WR[0] ;ERROR IF HERE
    INC WR[0] ;INCREMENT COUNTER
    RETURN+1
T2.SUB.18:
    INC WR[0] ;INCREMENT COUNTER
    RETURN+1
T2.SUB.1:
    JSR [T2.SUB.2] ;CALL NEXT SUBROUTINE IN NEST
    INC WR[0] ;INCREMENT THE COUNTER TO PROVE WE'VE BEEN HERE
    RETURN
T2.SUB.2:
    JSR [T2.SUB.3] ;CALL NEXT SUBROUTINE IN NEST
    INC WR[0] ;INCREMENT THE COUNTER TO PROVE WE'VE BEEN HERE
    RETURN
T2.SUB.3:
    JSR [T2.SUB.4] ;CALL NEXT SUBROUTINE IN NEST
    INC WR[0] ;INCREMENT THE COUNTER TO PROVE WE'VE BEEN HERE
    RETURN
T2.SUB.4:
    
```

U 154A, 8954,DC	:6470	JSR [T2.SUB.5]	:CALL NEXT SUBROUTINE IN NEST
U 154B, 2040,15	:6471	INC WR[0]	:INCREMENT THE COUNTER TO PROVE WE'VE BEEN HERE
U 154C, 5800,14	:6472	RETURN	
	:6473	T2.SUB.5:	
U 154D, 8955,0C	:6474	JSR [T2.SUB.6]	:CALL NEXT SUBROUTINE IN NEST
U 154E, 2040,15	:6475	INC WR[0]	:INCREMENT THE COUNTER TO PROVE WE'VE BEEN HERE
L 154F, 5800,14	:6476	RETURN	
	:6477	T2.SUB.6:	
U 1550, 8955,3C	:6478	JSR [T2.SUB.7]	:CALL NEXT SUBROUTINE IN NEST
U 1551, 2040,15	:6479	INC WR[0]	:INCREMENT THE COUNTER TO PROVE WE'VE BEEN HERE
U 1552, 5800,14	:6480	RETURN	
	:6481	T2.SUB.7:	
U 1553, 8955,6C	:6482	JSR [T2.SUB.8]	:CALL NEXT SUBROUTINE IN NEST
U 1554, 2040,15	:6483	INC WR[0]	:INCREMENT THE COUNTER TO PROVE WE'VE BEEN HERE
U 1555, 5800,14	:6484	RETURN	
	:6485	T2.SUB.8:	
U 1556, 8955,9C	:6486	JSR [T2.SUB.9]	:CALL NEXT SUBROUTINE IN NEST
U 1557, 2040,15	:6487	INC WR[0]	:INCREMENT THE COUNTER TO PROVE WE'VE BEEN HERE
U 1558, 5800,14	:6488	RETURN	
	:6489	T2.SUB.9:	
U 1559, 8955,CC	:6490	JSR [T2.SUB.10]	:CALL NEXT SUBROUTINE IN NEST
U 155A, 2040,15	:6491	INC WR[0]	:INCREMENT THE COUNTER TO PROVE WE'VE BEEN HERE
U 155B, 5800,14	:6492	RETURN	
	:6493	T2.SUB.10:	
U 155C, 8955,FC	:6494	JSR [T2.SUB.11]	:CALL NEXT SUBROUTINE IN NEST
U 155D, 2040,15	:6495	INC WR[0]	:INCREMENT THE COUNTER TO PROVE WE'VE BEEN HERE
U 155E, 5800,14	:6496	RETURN	
	:6497	T2.SUB.11:	
U 155F, 0956,2C	:6498	JSR [T2.SUB.12]	:CALL NEXT SUBROUTINE IN NEST
U 1560, 2040,15	:6499	INC WR[0]	:INCREMENT THE COUNTER TO PROVE WE'VE BEEN HERE
U 1561, 5800,14	:6500	RETURN	
	:6501	T2.SUB.12:	
U 1562, 8956,5C	:6502	JSR [T2.SUB.13]	:CALL NEXT SUBROUTINE IN NEST
U 1563, 2040,15	:6503	INC WR[0]	:INCREMENT THE COUNTER TO PROVE WE'VE BEEN HERE
U 1564, 5800,14	:6504	RETURN	
	:6505	T2.SUB.13:	
U 1565, 0956,8C	:6506	JSR [T2.SUB.14]	:CALL NEXT SUBROUTINE IN NEST
U 1566, 2040,15	:6507	INC WR[0]	:INCREMENT THE COUNTER TO PROVE WE'VE BEEN HERE
U 1567, 5800,14	:6508	RETURN	
	:6509	T2.SUB.14:	
U 1568, 0AA8,0C	:6510	JSR [T2.SUB.15]	:CALL NEXT SUBROUTINE IN NEST
U 1569, 2040,15	:6511	INC WR[0]	:INCREMENT THE COUNTER TO PROVE WE'VE BEEN HERE
U 156A, 5800,14	:6512	RETURN	
	:6513	2A80:	
	:6514	T2.SUB.15:	
U 2A80, 8AA8,4C	:6515	JSR [T2.SUB.16]	:CALL NEXT SUBROUTINE IN NEST
U 2A81, 2040,15	:6516	INC WR[0]	:INCREMENT THE COUNTER TO PROVE WE'VE BEEN HERE
U 2A82, 2040,15	:6517	INC WR[0]	
U 2A83, 5800,14	:6518	RETURN	
	:6519	T2.SUB.16:	
U 2A84, 2040,15	:6520	INC WR[0]	:INCREMENT THE COUNTER TO PROVE WE'VE BEEN HERE
U 2A85, 5800,14	:6521	RETURN	
	:6522	156F:	
	:6523	END.T2:	

Page "TEST 3 - Loop control (jump to stack) Tests (M8390 CPU Module)"

++

Test Description:

This test checks the JMP.Z.CLR and JMP.C.SET skip control functions. These instructions do a jump to the top entry of the stack if the condition is met and do not pop the stack. A loop can then be performed until the condition goes away. When that happens the stack is popped to remove the entry but the u-sequencer goes to the next sequential address. The test loads the ALU CC's with 0's, does 3 JSR's to get 3 addresses on the stack, and executes a JMP.Z.CLR. This will cause the u-code to loop back to the last JSR location + 1 where checks are made. The ALU Z bit is set via the Y BUS, and JMP.Z.CLR is executed again. Since ALU Z is now set, the JMP.Z.CLR should pop the stack and fall through to the next instruction. This test will detect if the loop fails to occur, or if the loop pops the stack, or if the stack is not popped after the condition is met. JMP.C.SET is tested in a similar manner.

Assumptions:

It is assumed that all the previous JSR, RETURN and stack tests have run successfully.

Test Steps:

- 1) Set up error mask, error number, and module number in LS (for error printouts) and clear previous error number in LS.
- 2) Clear all ALU CC's and execute 2 JSR's.
- 3) At the second JSR destination execute another JSR to the JMP.Z.CLR instruction.
- 4) The JMP.Z.CLR will loop back to the last JSR location + 1 where Z is checked. If Z is clear, the u-code will set Z and execute the JMP.Z.CLR again. Otherwise an error is reported (JMP.Z.CLR looped with Z set).
- 5) Now that Z is set, the JMP.Z.CLR should pop the stack and fall through to the next instruction. If JMP.Z.CLR loops again, the error will be detected in step 4.
- 6) If JMP.Z.CLR falls through, the Z bit is checked. If Z is not set, an error is reported (JMP.Z.CLR failed to loop).
- 7) The error number is changed and a RETURN is executed. If the pop worked, the RETURN will go to the second JSR location + 1 and the JMP.Z.CLR test is complete.
- 8) Set all ALU CC's and execute 2 JSR's.
- 9) At the second JSR destination execute another JSR to the JMP.C.SET instruction.
- 10) The JMP.C.SET will loop back to the last JSR location + 1 where C is checked. If C is set, the u-code will clear C and execute the JMP.C.SET again. Otherwise an error is reported (JMP.C.SET looped with C clear).
- 11) Now that C is clear, the JMP.C.SET should pop the stack and fall through to the next instruction. If JMP.C.SET loops again, the error will be detected in step 10.
- 12) If JMP.C.SET falls through, the C bit is checked. If C is not

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

M 12
TEST 3 - Loop contr 3-MAR-82
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module
TEST 3 - Loop control (jump to stack) Tests (M8390 CPU Module)

Fiche 1 Frame M12
Sequence 155
Rev 01.00 Page 149

:6579 : clear, an error is reported (JMP.C.SET failed to loop).
:6580 : 13) The error number is changed and a RETURN is executed. If the pop
:6581 : worked, the RETURN will go to the second JSR location + 1 and the
:6582 : test is complete.
:6583 :
:6584 :
:6585 :
:6586 :
:6587 :
:6588 :
:6589 :
:6590 :
:6591 :
:6592 :
:6593 :
:6594 :
:6595 :
:6596 :
:6597 :
:6598 :
:6599 :
:6600 :
:6601 :
:6602 :
:6603 :
:6604 :
:6605 :
:6606 :
:6607 :
:6608 :
:6609 :
:6610 :
:6611 :
:6612 :
:6613 :
:6614 :
:6615 :
:6616 :
:6617 :
:6618 :
:6619 :
:6620 :
:6621 :
:6622 :
:6623 :
:6624 :
:6625 :
:6626 :
:6627 :
:6628 :
:6629 :
:6630 :
:6631 :
:6632 :
:6633 :

Errors:

- Error 1 - JMP.Z.CLR failed to loop.
- Error 2 - JMP.Z.CLR looped with Z set.
- Error 3 - JMP.Z.CLR failed to pop the stack when it fell through.
- Error 4 - JMP.Z.CLR popped the stack during the loop.
- Error 5 - JMP.C.SET failed to loop.
- Error 6 - JMP.C.SET looped with C clear.
- Error 7 - JMP.C.SET failed to pop the stack when it fell through.
- Error 8 - JMP.C.SET popped the stack during the loop.

Debug:

- Error 1 - This error indicates that ENABLE RTS L did not go low when the JMP.Z.CLR was executed with Z clear. The most likely suspect is the SEQ.CTL PAL itself. All the inputs to this PAL have been used previously. The inputs can be checked if desired for the following: CSR 4-0 should equal 11(H), and pin 3 should be low. The output on pin 17 should be high. These inputs should be checked during the u-word JMP.Z.CLR when Z is clear (the first execution of JMP.Z.CLR).
- Error 2 - This error indicates that ENABLE RTS L did not go high when the JMP.Z.CLR was executed with Z set. The most likely suspect is the SEQ.CTL PAL itself. All the inputs to this PAL have been used previously. The inputs can be checked if desired for the following: CSR 4-0 should equal 11(H), and pin 3 should be high. The output on pin 17 should be low. These inputs should be checked during the u-word JMP.Z.CLR when Z is set (the second execution of JMP.Z.CLR).
- Error 3 - This error indicates that ENABLE SP L did not go low when the JMP.Z.CLR was executed with Z set. The most likely suspect is the SEQ.CTL PAL itself. All the inputs to this PAL have been used previously. The inputs can be checked if desired for the following: CSR 4-0 should equal 11(H), and pin 3 should be high. The output ENABLE SP L on pin 15 should be low. These inputs should be checked during the u-word JMP.Z.CLR when Z is set (the second execution of JMP.Z.CLR).
- Error 4 - This error indicates that ENABLE SP L did not go high when the JMP.Z.CLR was executed with Z clear. The most likely suspect is the SEQ.CTL PAL itself. All the inputs to this PAL have been used previously. The inputs can be checked if desired for the following: CSR 4-0 should equal 11(H), and pin 3 should be low. The output ENABLE SP L on pin 15 should be high. These inputs should be checked during the u-word JMP.Z.CLR when Z is clear (the first execution of JMP.Z.CLR).
- Error 5 - This error indicates that ENABLE RTS L did not go low when the JMP.C.SET was executed with C set. The most likely suspect is the SEQ.CTL PAL itself. All the inputs to this PAL have been used

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

TEST 3 - Loop contr 3-MAR-82 N 12
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module
TEST 3 - Loop control (jump to stack) Tests (M8390 CPU Module) Fiche 1 Frame N12
Sequence 156
Rev 01.00 Page 150

:6634 : previously. The inputs can be checked if desired for the following:
:6635 : CSR 4-0 should equal 13(H), and pin 3 should be low. The output on
:6636 : pin 17 should be high. These inputs should be checked during the u-
:6637 : word JMP.C.SET when C is set (the first execution of JMP.C.SET).
:6638 :
:6639 : Error 6 - This error indicates that ENABLE RTS L did not go high when
:6640 : the JMP.C.SET was executed with C clear. The most likely suspect is
:6641 : the SEQ.CTL PAL itself. All the inputs to this PAL have been used
:6642 : previously. The inputs can be checked if desired for the following:
:6643 : CSR 4-0 should equal 13(H), and pin 3 should be high. The output on
:6644 : pin 17 should be low. These inputs should be checked during the u-
:6645 : word JMP.C.SET when C is clear (the second execution of JMP.C.SET).
:6646 :
:6647 : Error 7 - This error indicates that ENABLE SP L did not go low when
:6648 : the JMP.C.SET was executed with C clear. The most likely suspect is
:6649 : the SEQ.CTL PAL itself. All the inputs to this PAL have been used
:6650 : previously. The inputs can be checked if desired for the following:
:6651 : CSR 4-0 should equal 13(H), and pin 3 should be high. The output
:6652 : ENABLE SP L on pin 15 should be low. These inputs should be checked
:6653 : during the u-word JMP.C.SET when C is clear (the second execution of
:6654 : JMP.C.SET).
:6655 :
:6656 : Error 8 - This error indicates that ENABLE SP L did not go high when
:6657 : the JMP.C.SET was executed with C set. The most likely suspect is
:6658 : the SEQ.CTL PAL itself. All the inputs to this PAL have been used
:6659 : previously. The inputs can be checked if desired for the following:
:6660 : CSR 4-0 should equal 13(H), and pin 3 should be low. The output
:6661 : ENABLE SP L on pin 15 should be high. These inputs should be checked
:6662 : during the u-word JMP.C.SET when C is set (the first execution of
:6663 : JMP.C.SET).
:6664 :
:6665 :
:6666 :
:6667 :
:6668 :
:6669 :
:6670 :
:6671 :
:6672 :
:6673 :
:6674 :
:6675 :
:6676 :
:6677 :
:6678 :
:6679 :
:6680 :
:6681 :
:6682 :
:6683 :
:6684 :
:6685 :
:6686 :
:6687 :
:6688 :

U 156F, B65E,15 :6667
U 1570, 3E80,15 :6668
U 1571, 10E0,15 :6669
U 1572, 0957,24 :6670
U 1573, E58A,15 :6671
U 1574, 65A0,15 :6672
U 1575, 2F80,15 :6673
U 1576, 2040,15 :6674
U 1577, BE82,15 :6675
U 1578, A3C0,15 :6676
U 1579, 3E8C,15 :6677
U 157A, C76E,15 :6678
U 157B, 3E80,15 :6679
U 157C, 2F80,15 :6680
U 157D, 3FFC,15 :6681
U 157E, 0958,6C :6682
U 157F, 3644,15 :6683
U 1580, BE82,15 :6684
U 1581, 2F80,15 :6685

T.3:--
T.3: MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT15 IN WRO FOR CONTROL AND STATUS WORD
MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT15 IN CNTROL AND STATUS WORD
MISC [SET.CP.ATTN] ;BIT15 INDICATES START OF TEST TO CONSOLE
;ISSUE CPU ATTN TO CONSOLE
WAIT.T3.0: JMP [WAIT.T3.0] ;LOOP HERE WAITING FOR CONSOLE TO RESPOND
CLR LS[ERROR.MASK] ;CLEAR ERROR MASK.
CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
INC WR[0] ;SET WRO TO 1
MOV WR[0] TO LS[ERROR.NUMBER] ;SET ERROR NUMBER TO FIRST ERROR
ROL WR[0] ;SET WRO TO 2
MOV WR[0] TO LS[MODULE.NUM] ;SET UP MODULE CODE FOR CPU MODULE
BIS LS[BIT23] TO WR[0]
MOV WR[0] TO LS[CONTROL.STATUS] ;LOAD CONTROL AND STATUS WORD
T3.BEGIN: CLR WR[0]
MOV WR[0] TO LS[ALU.CC] ;CLEAR THE ALU CONDITION CODES
JSR [SUB.T3.A] ;JUMP TO FIRST OF THREE NESTED SUBROUTINES
MOV LS[#4] TO WR[0]
MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR NUMBER = 4
CLR WR[0] ;SET DELIBERATELY ERRONEOUS RESULTS IN

```

U 1582, 369E,95 :6689      MOV LS[ONES] TO WR[1]          ; IN WRO & 1 TO FLAG LOOP ERROR
U 1583, 0869,3C :6690      JSR [CHECK.RESULT]
U 1584, 8957,C4 :6691      JMP [T3.BEGIN]                ; LOOP ON ERROR IF ENABLED
U 1585, 895A,24 :6692      JMP [END.T3]                  ; GO TO END OF TEST
                                :6693
SUB.T3.A:
U 1586, 8958,8C :6694      JSR [SUB.T3.B]                ; JUMP TO SECOND OF THREE NESTED SUBROUTINES
L 1587, 895A,24 :6695      JMP [END.T3]                  ; JUMP TO END
                                :6696
SUB.T3.B:
U 1588, 0959,2C :6697      JSR [SUB.T3.C]                ; JUMP TO LAST OF THREE NESTED SUBROUTINES
U 1589, 0959,C1 :6698      JMP.IF[NEQ] TO [T3.D]          ; JUMP IF Z CLEAR
U 158A, 2F80,15 :6699      CLR WR[0]                    ; SET DELIBERATELY ERRONEOUS DATA IN
U 158B, 369E,95 :6700      MOV LS[ONES] TO WR[1]          ; WRO & 1 TO FLAG LOOP ERROR
U 158C, 0869,3C :6701      JSR [CHECK.RESULT]
U 158D, 8957,C4 :6702      JMP [T3.BEGIN]                ; LOOP ON ERROR IF ENABLED
U 158E, 3682,15 :6703      MOV LS[ERROR.NUMBER] TO WR[0] ; GET ERROR NUMBER
                                :6704
U 158F, CD42,35 :6705      XOR LS[#2] WITH WR[0] TO 0,    ; IS ERROR NUMBER = 2 ?
U 1590, 895A,09 :6706      DT(LONG)&SET.ALU.CC
U 1591, 5B00,14 :6707      JMP.IF[EQL] TO [T3.E]          ; JUMP IF SO
                                :6708
SUB.T3.C:
U 1592, 5B00,11 :6709      SCTL/JMP.Z.CLR
U 1593, 895A,09 :6710      JMP.IF[EQL] TO [T3.E]          ; JUMP IF Z SET
U 1594, 2F80,15 :6711      CLR WR[0]
U 1595, 2040,15 :6712      INC WR[0]
U 1596, BE82,15 :6713      MOV WR[0] TO LS[ERROR.NUMBER] ; ERROR NUMBER = 1
U 1597, 2F80,15 :6714      CLR WR[0]                    ; SET DELIBERATELY ERRONEOUS DATA IN
U 1598, 369E,95 :6715      MOV LS[ONES] TO WR[1]          ; WRO & 1 TO FLAG LOOP ERROR
U 1599, 0869,3C :6716      JSR [CHECK.RESULT]
U 159A, 8957,C4 :6717      JMP [T3.BEGIN]                ; LOOP ON ERROR IF ENABLED
U 159B, 095A,04 :6718      JMP [T3.E]
                                :6719
T3.D:
U 159C, 8870,0C :6720      JSR [INC.EN]                  ; INCREMENT ERROR NUMBER
U 159D, 3644,15 :6721      MOV LS[BIT2] TO WR[0]
U 159E, 3FFC,15 :6722      MOV WR[0] TO LS[ALU.CC]        ; SET Z BIT
U 159F, 8959,24 :6723      JMP [SUB.T3.C]
                                :6724
T3.E:
U 15A0, 8870,0C :6725      JSR [INC.EN]                  ; INCREMENT ERROR NUMBER
U 15A1, 5B00,14 :6726      RETURN
                                :6727
END.T3:
  
```

Page 'TEST 4 - OS 'ORed' with NAD 0-4 (M8390 CPU Module)'

Test Description:

This test checks the logic that 'ORs' OS 0 H through OS 4 H with the lower 5 bits of the jump address during a JUMP instruction with CSR 18 set. The test will load OS with data of all 0's or data of all 1's and execute a JSR with the lower 5 bits of the jump address (CSR 4 through 8) either all set or all clear. The JSR instruction will jump to a block of 32 u-words containing INC WR 0 instructions. After this block of INC instructions is a RETURN. An error is detected by checking WR 0 after the return. It should be 20(H) if the jump is with OS of all zeros and CSR 4 through 8 of all zeros. The other three data patterns should jump to the last word in the block and WR 0 will contain a 1.

Assumptions:

It is assumed that the previous tests have been run successfully.

Test Steps:

- 1) Set up error mask, error number, and module number in LS (for error printouts) and clear previous error number in LS.
- 2) Clear WR 0 and load the OS reg with all zeros.
- 3) Execute a JSR instruction with the lower 5 bits of the jump address=0. The JSR is to a block of 32 INC instructions followed by a RETURN.
- 4) After the RETURN check WR 0 for 20(H).
- 5) Clear WR 0 and execute JSR with 1's in the lower 5 jump address bits.
- 6) After the return, check WR 0 for a 1(H).
- 7) Load OS with all 1's, clear WR 0, and repeat step 3. Check WR 0 for a 1(H).
- 8) Repeat steps 5 and 6 with OS containing all ones.

Errors:

- Error 1 - Jump with CSR 18 set ('OF' OS with jump address) failed with data of 0's in OS and 0's in CSR 4 through CSR 8.
- Error 2 - Jump with CSR 18 set ('OR' OS with jump address) failed with data of 0's in OS and 1's in CSR 4 through CSR 8.
- Error 3 - Jump with CSR 18 set ('JR' OS with jump address) failed with data of 1's in OS and 0's in CSR 4 through CSR 8.
- Error 4 - Jump with CSR 18 set ('OR' OS with jump address) failed with data of 1's in OS and 1's in CSR 4 through CSR 8.

Debug:

Error 1 - This error indicates a problem with the CSR.OS MUX PAL or its inputs. During the JSR instruction, check the inputs to the CSR.OS MUX PAL for a high on CSR 18 H and lows on all other inputs. If these are correct, the CSR.OS MUX PAL is probably bad (the outputs should be all 0's). The received data printed in the error report

ZZ-ENKCB-1.0 : ENKCB.MIC
: ENKCB.MCR
: ENKCB.MIC

TEST 4 - OS 'ORed' 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Rev 01.00
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Rev 01.00
TEST 4 - OS 'ORed' with NAD 0-4 (M8390 CPU Module)

Fiche 1 Frame D13

Sequence 159

Page 153

```
:6783 : tells how many INC instructions were executed so it can be determined
:6784 : where in the block of 32 INC u-words the jump went to. For example,
:6785 : if received data=1E, then the jump went to the third word in the 32
:6786 : word block.
:6787 :
:6788 : Errors 2-4 - See error 1. Refer to the error summary above for the
:6789 : inputs to the CSR.OS MUX PAL. The output should be all ones for all
:6790 : 3 errors.
:6791 :
:6792 :
:6793 :
:6794 :
:6795 :
:6796 :
:6797 :
:6798 :
:6799 :
:6800 :
:6801 :
:6802 :
:6803 :
:6804 :
:6805 :
:6806 :
:6807 :
:6808 :
:6809 :
:6810 :
:6811 :
:6812 :
:6813 :
:6814 :
:6815 :
:6816 :
:6817 :
:6818 :
:6819 :
:6820 :
:6821 :
:6822 :
:6823 :
:6824 :
:6825 :
:6826 :
:6827 :
:6828 :
:6829 :
:6830 :
:6831 :
:6832 :
:6833 :
:6834 :
:6835 :
:6836 :
:6837 :
```

T.4:--

U 15A2, B65E,15 :6794
U 15A3, 3E80,15 :6795
U 15A4, 10E0,15 :6796
U 15A5, 095A,54 :6797
U 15A6, E58A,15 :6798
U 15A7, 65A0,15 :6799
U 15A8, 2F80,15 :6800
U 15A9, 2040,15 :6801
U 15AA, BE82,15 :6802
U 15AB, A3C0,15 :6803
U 15AC, 3E8C,15 :6804
U 15AD, B664,15 :6805
U 15AE, 3E80,15 :6806
U 15AF, 2F80,15 :6807
U 15B0, 3EF8,15 :6808
U 15B1, 8D5E,0C :6809
U 15B2, 364A,95 :6810
U 15B3, 0869,3C :6811
U 15B4, 095A,F4 :6812
U 15B5, 8870,0C :6813
U 15B6, 2F80,15 :6814
U 15B7, 0D5F,FC :6815
U 15B8, 3640,95 :6816
U 15B9, 0869,3C :6817
U 15BA, 895B,64 :6818
U 15BB, 8870,0C :6819
U 15BC, 2F80,15 :6820
U 15BD, 369E,95 :6821
U 15BE, BEF8,95 :6822
U 15BF, 8D5E,0C :6823
U 15C0, 3640,95 :6824
U 15C1, 0869,3C :6825
U 15C2, 895B,C4 :6826
U 15C3, 8870,0C :6827
U 15C4, 2F80,15 :6828
U 15C5, 0D5F,FC :6829
U 15C6, 3640,95 :6830
U 15C7, 0869,3C :6831

MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT15 IN WRO FOR CONTROL AND STATUS WORD
MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT15 IN CNTROL AND STATUS WORD
MISC [SET.CP.ATTN] ;BIT15 INDICATES START OF TEST TO CONSOLE
ISSUE CPU ATTN TO CONSOLE

WAIT.T4.0:
JMP [WAIT.T4.0] ;LOOP HERE WAITING FOR CONSOLE TO RESPOND
CLR LS[ERROR.MASK] ;CLEAR ERROR MASK.
CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
CLR WR[0]
INC WR[0] ;SET WRO TO 1
MOV WR[0] TO LS[ERROR.NUMBER] ;SET ERROR NUMBER TO FIRST ERROR
ROL WR[0] ;SET WRO TO 2
MOV WR[0] TO LS[MODULE.NUM] ;SET UP MODULE CODE FOR CPU MODULE
MOV LS[BIT18] TO WR[0]
MOV WR[0] TO LS[CONTROL.STATUS] ;LOAD CONTROL AND STATUS WORD.

LOOP.T4.1:
CLR WR[0] ;WRO = 0
MOV WR[0] TO LS[OS] ;OS = 0
JSR.OS [T4.SUB.1] ;SUBROUTINE CALL
MOV LS[#20] TO WR[1] ;EXPECTED DATA = 00000020
JSR [CHECK.RESULT] ;CHECK WRO
JMP [LOOP.T4.1] ;LOOP ON ERROR IF ENABLED
JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 2

LOOP.T4.2:
CLR WR[0]
JSR.OS [T4.SUB.2] ;SUBROUTINE CALL
MOV LS[#1] TO WR[1] ;EXPECTED DATA = 1
JSR [CHECK.RESULT] ;CHECK WRO
JMP [LOOP.T4.2] ;LOOP ON ERROR IF ENABLED
JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 3

LOOP.T4.3:
CLR WR[0]
MOV LS[ONES] TO WR[1]
MOV WR[1] TO LS[OS] ;SET ALL BITS IN OS REGISTER
JSR.OS [T4.SUB.1] ;SUBROUTINE CALL
MOV LS[#1] TO WR[1] ;EXPECTED DATA = 1
JSR [CHECK.RESULT] ;CHECK WRO
JMP [LOOP.T4.3] ;LOOP ON ERROR IF ENABLED
JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 4

LOOP.T4.4:
CLR WR[0]
JSR.OS [T4.SUB.2] ;SUBROUTINE CALL
MOV LS[#1] TO WR[1] ;EXPECTED DATA = 1
JSR [CHECK.RESULT]

```

U 15C8, 895C,44 :6838      JMP [LOOP.T4.4]      ;LOOP ON ERROR IF ENABLED
U 15C9, 8960,14 :6839      JMP [END.T4]        ;ALL DONE
                  :6840
                  :6841
15E0:
T4.SUB.1:
U 15E0, 2040,15 :6842      INC WR[0]          ;BLOCK OF 32 INCREMENTS
U 15E1, 2040,15 :6843      INC WR[0]
U 15E2, 2040,15 :6844      INC WR[0]
U 15E3, 2040,15 :6845      INC WR[0]
U 15E4, 2040,15 :6846      INC WR[0]
U 15E5, 2040,15 :6847      INC WR[0]
U 15E6, 2040,15 :6848      INC WR[0]
U 15E7, 2040,15 :6849      INC WR[0]
U 15E8, 2040,15 :6850      INC WR[0]
U 15E9, 2040,15 :6851      INC WR[0]
U 15EA, 2040,15 :6852      INC WR[0]
U 15EB, 2040,15 :6853      INC WR[0]
U 15EC, 2040,15 :6854      INC WR[0]
U 15ED, 2040,15 :6855      INC WR[0]
U 15EE, 2040,15 :6856      INC WR[0]
U 15EF, 2040,15 :6857      INC WR[0]
U 15F0, 2040,15 :6858      INC WR[0]
U 15F1, 2040,15 :6859      INC WR[0]
U 15F2, 2040,15 :6860      INC WR[0]
U 15F3, 2040,15 :6861      INC WR[0]
U 15F4, 2040,15 :6862      INC WR[0]
U 15F5, 2040,15 :6863      INC WR[0]
U 15F6, 2040,15 :6864      INC WR[0]
U 15F7, 2040,15 :6865      INC WR[0]
U 15F8, 2040,15 :6866      INC WR[0]
U 15F9, 2040,15 :6867      INC WR[0]
U 15FA, 2040,15 :6868      INC WR[0]
U 15FB, 2040,15 :6869      INC WR[0]
U 15FC, 2040,15 :6870      INC WR[0]
U 15FD, 2040,15 :6871      INC WR[0]
U 15FE, 2040,15 :6872      INC WR[0]
                  :6873
T4.SUB.2:
U 15FF, 2040,15 :6874      INC WR[0]
U 1600, 5800,14 :6875      RETURN
                  :6876
END.T4:
  
```

:6877 Page "TEST 5 - CONS HALT, Skip and Jump on Interrupt (M8390 CPU Module)"
:6878 ++

:6879
:6880 Test Description:

:6881 This test checks the console generated signal CONS HALT and the skip
:6882 logic involved with the interrupt CONS HALT generates. It also tests
:6883 the identifier code on BUS D D02 through BUS D D05. A CPU ATTN is
:6884 issued to tell the console to assert CONS HALT when needed. The test
:6885 flow is as follows: A CPU ATTN is issued with bits 16 and 17 of the
:6886 control and status word set, indicating that the console should assert
:6887 CONS HALT. This signal is latched and sent to the INTR CTRL PAL. As
:6888 long as HALT is not masked by a MISC instruction, the PAL asserts IRQ
:6889 OUT L which goes through an 'OR' gate and is latched as INTERR REQ H.
:6890 This signal goes to the 8 to 1 mux in the skip logic. A skip if no
:6891 interrupt instruction is used to test this signal. The identifier
:6892 code is read by issuing a MOVE with LS address=7E. This enables the
:6893 output of the INTR CTRL PAL by asserting EN STAT REG L low. The test
:6894 is repeated with CONS HALT deasserted to test for a skip and the
:6895 identifier code is checked for 1111 (no interrupt pending). Finally,
:6896 JUMP on no interrupt is tested with an interrupt asserted and un-
:6897 asserted. Note: 3 u-words must be issued after the console returns
:6898 control to the U-code to latch everything up before the skip is
:6899 tested.
:6900

:6901
:6902 Assumptions:

:6903 It is assumed that the skip logic for other skips has been previously
:6904 tested.
:6905

:6906
:6907 Test Steps:

- :6908
:6909 1) Set up error mask, error number, and module number in LS (for
:6910 error printouts) and clear previous error number in LS.
:6911 2) Issue CPU ATTN with bits 16 and 17 of the control and status word
:6912 set to assert CONS HALT L.
:6913 3) Load the IPL with 1F(H) to block any other interrupts (takes 2 u-
:6914 words). Execute a 2 NOP's, then execute a MOVE with SCTL=NO.IN-
:6915 TERRUPT and LS=7E to read the identifier code and check for no
:6916 skip.
:6917 4) Check identifier code for 0 on BUS D D05.
:6918 5) Issue CPU ATTN with bit 16 set and 17 clear in the control and
:6919 status word to deassert CONS HALT L.
:6920 6) Execute 3 NOP's, then execute a MOVE with SCTL=NO.interrupt and
:6921 LS=7E to read the identifier code and check for a skip.
:6922 7) Check identifier code for 1 on BUS D D05.
:6923 8) Issue JUMP with JCTL=NO.interrupt. Check for a jump.
:6924 9) Repeat step 2, execute 3 NOP's, then repeat step 8, checking for
:6925 no jump.
:6926

:6927
:6928 Errors:

- :6929 Error 1 - SCTL=skip if no interrupt skipped with CONS HALT asserted.
:6930 Error 2 - CONS HALT asserted failed to produce 0000 identifier code.
:6931 Error 3 - SCTL=skip if no interrupt failed to skip with CONS HALT de-

```

:6932      asserted.
:6933      Error 4 - CONS HALT deasserted failed to produce 1111 identifier code.
:6934      Error 5 - JCTL=jump if no interrupt failed to jump with CONS HALT de-
:6935      asserted.
:6936      Error 6 - JCTL=jump if no interrupt jumped with CONS HALT asserted.
:6937
:6938      Debug:
:6939
:6940      Error 1 - This error could be caused by a number of problems. Fir t
:6941      check the signal IRQ OUT LOW on pin 19 of the INTR CTRL PAL. It
:6942      should be low during the MOVE instruction with SCTL=NO.interrupt. If
:6943      not check the inputs for a low on HALT L (pin 2) and a high on MASK L
:6944      (pin 18). HALT L can be traced back through the 8 bit latch which
:6945      feeds the PAL, to the 8 bit latch that is loaded by the console. If
:6946      CONS HALT L is high at the first latch, check the inputs during the
:6947      time that the console is writing this latch. If the IRQ OUT L signal
:6948      is OK, check the output of the 2 input negated 'OR' gate for a high
:6949      and a high coming out of the 8 bit latch producing INTERR REQ H. If
:6950      this is OK, check the signal as it goes into the 8 to 1 mux in the
:6951      skip control logic. During the SCTL=NO.interrupt instruction, the out-
:6952      put on pin 6 of this mux should be low. If not the 8 to 1 MUX is sus-
:6953      pect. If the signal is OK, the SEQ.CTL PAL is probably bad.
:6954
:6955      Error 2 - If this is the first error, the problem is either in the
:6956      INCR CTRL PAL or the signal EN STAT REG L. Check EN STAT REG L at
:6957      the INCR CTRL PAL during the MOVE instruction for a low. If this
:6958      is OK, the PAL itself is probably bad. The other inputs to this PAL
:6959      should be a low on HALT L (pin 2) and a high on MASK L (pin 18). If
:6960      4EN STAT REG L is high, check the output at the REG CONTROL PAL. If
:6961      high here, the PAL is probably bad.
:6962
:6963      Error 3 - Same as error 1 except INTERR REQ H should be low and all
:6964      signals associated with CONS HALT should be unasserted.
:6965
:6966      Error 4 - Suspect the INTR CTRL PAL.
:6967
:6968      Error 5 - Suspect the SEQ.CTL PAL.
:6969
:6970      Error 6 - Suspect the SEQ.CTL PAL.
:6971
:6972      --
:6973
:U 1601, B65E,15 :6974      T.5:      MOV LS[BEGIN.TEST] TO WR[0]      ;SET BIT15 IN WRO FOR CONTROL AND STATUS WORD
:U 1602, 3E80,15 :6975      MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT15 IN CNTROL AND STATUS WORD
:6976      ; BIT15 INDICATES START OF TEST TO CONSOLE
:U 1603, 10E0,15 :6977      MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:6978
:U 1604, 8960,44 :6979      WAIT.T5.0:      JMP [WAIT.T5.0] ;LOOP HERE WAITING FOR CONSOLE TO RESPOND
:U 1605, B64A,15 :6980      MOV LS[BIT5] TO WR[0]
:U 1606, F88A,15 :6981      MCOM WR[0] TO LS[ERROR.MASK] ;ERROR MASK = #FFFFFFDF
:U 1607, 65A0,15 :6982      CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:U 1608, 3642,15 :6983      MOV LS[CPU] TO WR[0] ;PUT A 2 IN WR 0
:U 1609, 3E8C,15 :6984      MOV WR[0] TO LS[MODULE.NUM] ;SET UP MODULE CODE FOR CPU MODULE
:6985
:U 160A, 0870,8C :6986      LOOP.T5.1:      JSR [ASSERT.NA]
  
```

```

U 160B, B640,15 :6987      MOV LS[#1] TO WR[0]          ;PUT A 1 IN WR 0
U 160C, BE82,15 :6988      MOV WR[0] TO LS[ERROR.NUMBER] ;SET ERROR NUMBER TO FIRST ERROR
U 160D, 3660,15 :6989      MOV LS[INTERUPT.EN] TO WR[0]
U 160E, C762,15 :6990      BIS LS[CONSOLE.HALT] TO WR[0]
U 160F, 3E80,15 :6991      MOV WR[0] TO LS[CONTROL.STATUS] ;SET UP TO SET CONSOLE HALT
U 1610, 10E0,15 :6992      MISC [SET.CP.ATTN]          ;GET CONSOLE'S ATTENTION
                        :6993
WAIT.T5.1:
U 1611, 0961,14 :6994      JMP [WAIT.T5.1]            ;LOOP UNTIL CONSOLE RESPONDS
U 1612, B724,15 :6995      MOV LS[HI.IPL] TO WR[0]    ;SET BITS 16-20 IN WR
U 1613, BFFE,15 :6996      MOV WR[0] TO LS[PSL.HW]    ;IPL = 1F
U 1614, DB00,15 :6997      NOP
U 1615, DB00,15 :6998      NOP                        ;ALLOW TIME TO TAKE EFFECT
U 1616, 36FD,95 :6999      MOV LS[INTERUPT.VEC] TO WR[3] ;GET IDENTIFIER CODE
U 1617, DB00,07 :7000      SKIP.IF[NO.INTERUPT]        ;TEST FOR SKIP
U 1618, 0961,04 :7001      JMP [T5.2]                ;GO TO NEXT
U 1619, 2F80,15 :7002      CLR WR[0]                 ;SKIPPED! --ERROR--
U 161A, 369E,95 :7003      MOV LS[ONES] TO WR[1]      ;SET ERRONEOUS DATA
U 161B, 0869,3C :7004      JSR [CHECK.RESULT]         ;CALL ERROR ROUTINE
U 161C, 0960,A4 :7005      JMP [LOOP.T5.1]            ;LOOP ON ERROR IF ENABLED
                        :7006
T5.2:
U 161D, 8870,0C :7007      JSR [INC.EN]                ;INCREMENT ERROR NUMBER TO 2
U 161E, 0870,BC :7008      JSR [DEASSERT.NA]
U 161F, A006,15 :7009      MOV WR[3] TO WR[0]          ;GET IDENTIFIER CODE
U 1620, 2F82,95 :7010      CLR WR[1]                 ;EXPECTED DATA = 0
U 1621, 0869,3C :7011      JSR [CHECK.RESULT]         ;CHECK IDENTIFIER CODE
U 1622, 0960,A4 :7012      JMP [LOOP.T5.1]            ;LOOP ON ERROR IF ENABLED
                        :7013
LOOP.T5.3:
U 1623, 0870,8C :7014      JSR [ASSERT.NA]
U 1624, 36C0,15 :7015      MOV LS[#3(H)] TO WR[0]      ;PUT A 3 IN WR
U 1625, BE82,15 :7016      MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR NUMBER = 3
U 1626, 3660,15 :7017      MOV LS[INTERUPT.EN] TO WR[0] ;CLEAR INTERRUPT
U 1627, 3E80,15 :7018      MOV WR[0] TO LS[CONTROL.STATUS] ;CSR = BIT16
U 1628, 10E0,15 :7019      MISC [SET.CP.ATTN]          ;GET CONSOLES ATTENTION
                        :7020
WAIT.T5.3:
U 1629, 8962,94 :7021      JMP [WAIT.T5.3]            ;WAIT FOR CONSOLE TO RESPOND
U 162A, DB00,15 :7022      NOP
U 162B, DB00,15 :7023      NOP
U 162C, DB00,15 :7024      NOP
U 162D, 36FD,95 :7025      MOV LS[INTERUPT.VEC] TO WR[3] ;GET IDENTIFIER CODE
U 162E, DB00,07 :7026      SKIP.IF[NO.INTERUPT]        ;CHECK SKIP
U 162F, 5B00,1E :7027      SKIP
U 1630, 0963,54 :7028      JMP [T5.4]                ;JUMP IF SKIP OCCURED
U 1631, 2F80,15 :7029      CLR WR[0]                 ;DID NOT SKIP! --ERROR--
U 1632, 369E,95 :7030      MOV LS[ONES] TO WR[1]      ;SET ERRONEOUS DATA
U 1633, 0869,3C :7031      JSR [CHECK.RESULT]         ;CALL ERROR ROUTINE
U 1634, 8962,34 :7032      JMP [LOOP.T5.3]            ;LOOP ON ERROR IF ENABLED
                        :7033
T5.4:
U 1635, 8870,0C :7034      JSR [INC.EN]                ;INCREMENT ERROR NUMBER TO 4
U 1636, 0870,BC :7035      JSR [DEASSERT.NA]
U 1637, A006,15 :7036      MOV WR[3] TO WR[0]          ;GET IDENTIFIER CODE
U 1638, 364A,95 :7037      MOV LS[BIT5] TO WR[1]      ;EXPECTED DATA = SET
U 1639, 0869,3C :7038      JSR [CHECK.RESULT]         ;CHECK IDENTIFIER CODE
U 163A, 8962,34 :7039      JMP [LOOP.T5.3]            ;LOOP ON ERROR IF ENABLED
U 163B, 8870,0C :7040      JSR [INC.EN]                ;INCREMENT ERROR NUMBER TO 5
                        :7041
LOOP.T5.5:

```

I 13

ZZ-ENKCB-1.0 : ENKCB.MIC TEST 5 - CONS HALT, 3-MAR-82 Fiche 1 Frame I13 Sequence 164
 : ENKCB.MCR MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Rev 01.00 Page 158
 : ENKCB.MIC TEST 5 - CONS HALT, Skip and Jump on Interrupt (M8390 CPU Module)

```

U 163C, 0870,8C :7042      JSR [ASSERT.NA]
U 163D, 0964,27 :7043      JMP.IF[NO.INTERRUPT] TO [T5.OK.5]
U 163E, 2F80,15 :7044      CLR WR[0] ;DID NOT JUMP! --ERROR--
U 163F, 369E,95 :7045      MOV LS[ONES] TO WR[1] ;SET ERRONEOUS DATA
U 1640, 0869,3C :7046      JSR [CHECK.RESULT] ;CALL ERROR ROUTINE
U 1641, 0963,C4 :7047      JMP [LOOP.T5.5] ;LOOP ON ERROR IF ENABLED
                        :7048
T5.OK.5:
U 1642, 3660,15 :7049      MOV LS[INTERUPT.EN] TO WP[0]
U 1643, C762,15 :7050      BIS LS[CONSOLE.HALT] TO WR[0] ;SET CONSOLE.HALT
U 1644, 3E80,15 :7051      MOV WR[0] TO LS[CONTROL.STATUS] ;CSR = BIT16:BIT17
U 1645, 10E0,15 :7052      MISC [SET.CP.ATTN] ;GET CONSOLE'S ATTENTION
                        :7053
WAIT.T5.5:
U 1646, 8964,64 :7054      JMP [WAIT.T5.5] ;LOOP UNTIL CONSOLE RESPONDS
U 1647, 8870,0C :7055      JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 6
                        :7056
LOOP.T5.6:
U 1648, 8964,A7 :7057      JMP.IF[NO.INTERRUPT] TO [T5.ERR.6]
U 1649, 0964,E4 :7058      JMP [CLEANUP.T5] ;ALL DONE
                        :7059
T5.ERR.6:
U 164A, 2F80,15 :7060      CLR WR[0] ;JUMPED! --ERROR--
U 164B, B69E,15 :7061      MOV LS[ONES] TO WR[0] ;SET ERRONEOUS DATA
U 164C, 0869,3C :7062      JSR [CHECK.RESULT] ;CALL ERROR ROUTINE
U 164D, 0964,84 :7063      JMP [LOOP.T5.6] ;LOOP ON ERROR IF ENABLED
                        :7064
CLEANUP.T5:
U 164E, 3660,15 :7065      MOV LS[INTERUPT.EN] TO WR[0]
U 164F, 3E80,15 :7066      MOV WR[0] TO LS[CONTROL.STATUS]
U 1650, 10E0,15 :7067      MISC [SET.CP.ATTN] ;CLEAR SIGNALS
                        :7068
WAIT.T5.6:
U 1651, 8965,14 :7069      JMP [WAIT.T5.6] ;WAIT FOR CONSOLE TO RESPOND
                        :7070
END.T5:
  
```

:7071 Page "TEST 6 - INTR CTLR PAL, IPL, and BUS IRQ Test (M8390 CPU Module)"
:7072 ++
:7073
:7074 Test Description:
:7075
:7076 This test asserts all the interrupt signals, one at a time, and checks
:7077 that they are enabled to assert INTERR REQ if the IPL is at the cor-
:7078 rect level or below. The interrupt signals PWR FAIL INT L, INTRVL TIM
:7079 INT L, and CONS ATTN are asserted by issuing a CPU ATTN to the console
:7080 and setting appropriate bits in the control and status word. The con-
:7081 sole will set the appropriate signals and return control to the u-
:7082 code. The BUS IRQ signals (BR levels 4 to 7) are controlled by issue-
:7083 ing a MISC instruction after loading OS with the desired value of BUS
:7084 IRQ 4 H through BUS IRQ 7 H in OS bits 0 through 3. The IPL bits are
:7085 loaded from the Y BUS via a MOVE instruction with LS=FF. This asserts
:7086 LOAD PSL L from the REG CONTROL PAL to clock the PSL latch. The test
:7087 will assert one of the interrupt lines and run a count pattern through
:7088 the IPL bits, starting at 1F(H), until the point at which the
:7089 interrupt should be allowed through. During this time, skip if no
:7090 interrupt will be checking for no interrupt. When the IPL code
:7091 reaches the value that allows the interrupt to assert a INTERR REQ,
:7092 skip if no interrupt should not skip. At this time we check the
:7093 identifier code for the correct value and continue the IPL count
:7094 pattern down to 0 as we check. Note: 3 u-words must be issued after
:7095 any call to the console to latch everything up. 2 u-words must be
:7096 issued after changing any BR lines to allow skip and 1 u-word must be
:7097 issued to check the identifier code.
:7098
:7099 Assumptions:
:7100
:7101 It is assumed that the previous interrupt test using CONSOLE HALT has
:7102 run successfully.
:7103
:7104 Test Steps:
:7105
:7106 1) Set up error mask, error number, and module number in LS (for
:7107 error printouts) and clear previous error number in LS.
:7108 2) Execute CPU ATTN with bits 16 and 18 set in the control and status
:7109 word to assert PWR FAIL INT L. The console will return control to
:7110 the next u-word.
:7111 3) Load WR 2 with 1F 00 00(H) (1F in BUS Y D20 to BUS Y D16), and
:7112 WR 3 with 1D 00 00(H). 1D is the first IPL value that should
:7113 allow the interrupt.
:7114 4) Execute MOVE with LS=FF from WR 2 to load the PSL IPL bits with
:7115 data (initially 1F(H)).
:7116 5) Execute CMP of WR 2 with WR 3, skip if WR 2 is not equal to WR 3.
:7117 Go to step 9 if WR 2 is equal to WR 3.
:7118 6) Execute a NOP to clock latch for INTERR REQ H.
:7119 7) Issue NOP with SCTL=skip if no interrupt and check for skip.
:7120 8) Subtract 1 00 00(H) from WR 2 to decrement IPL value by 1 and go
:7121 to step 4.
:7122 9) Execute NOP with SCTL=skip if no interrupt. Check for no skip.
:7123 10) Execute MOVE from LS 7E(H) to WR 0 to load identifier code into
:7124 WR 0, load WR 1 with expected data in bits 5 - 2.
:7125 11) Check identifier code, then test WR 2 for zero. If zero, go to

```

:7126      step 14, else continue.
:7127      12) Subtract 1 00 00(H) from WR 2 to decrement IPL value by 1.
:7128      13) Execute MOVE with LS=FF from WR 2 to load the PSL IPL bits with
:7129      data. Execute a NOP. Go to step 10.
:7130      14) Execute CPU ATTN with bits 16 and 19 set in the control and status
:7131      word to assert INTRVL TIM INT L. The console will return control
:7132      to the next u-word.
:7133      15) Load WR 2 with 1F 00 00(H) (1F in BUS Y D20 to BUS Y D16), and
:7134      WR 3 with 17 00 00(H). 17 is the first IPL value that should
:7135      allow the interrupt. Repeat steps 4 through 13 via JSR (steps 4
:7136      through 13 is a subroutine). Return to next step when done.
:7137      16) Execute CPU ATTN with bits 16 and 20 set in the control and status
:7138      word to assert CONS ATTN L. The console will return control
:7139      to the next u-word.
:7140      17) Load WR 2 with 1F 00 00(H) (1F in BUS Y D20 to BUS Y D16), and
:7141      WR 3 with 13 00 00(H). 13 is the first IPL value that should
:7142      allow the interrupt. Repeat steps 4 through 13 via JSR (steps 4
:7143      through 13 is a subroutine). Return to next step when done.
:7144      18) Execute CPU ATTN with bit 16 set in the control and status word
:7145      to deassert console interrupts.
:7146      19) Load WR 2 with 1F 00 00(H) (1F in BUS Y D20 to BUS Y D16), and
:7147      WR 3 with 16 00 00(H). 16 is the first IPL value that should
:7148      allow the interrupt with BR7. Repeat steps 4 through 13 via JSR
:7149      (steps 4 through 13 is a subroutine) with the following additions:
:7150      Step 4A: Load the OS reg with the BUS IRQ value desired in bits
:7151      3-0. Execute MISC instruction to load OS into BUS IRQ.
:7152      Step 9A: Load the OS reg with the BUS IRQ value desired in bits
:7153      3-0. Execute MISC instruction to load OS into BUS IRQ. Execute a
:7154      NOP.
:7155      Step 13: Eliminate NOP and change "go to step 10" to read "go to
:7156      step 9A".
:7157      Return to next step when done.
:7158      20) Repeat step 19 with data of 15 in WR 3, checking BR6.
:7159      21) Repeat step 19 with data of 14 in WR 3, checking BR5.
:7160      22) Repeat step 19 with data of 13 in WR 3, checking BR4.
:7161
:7162      Errors:
:7163
:7164      OTHER data : Interrupt Priority Level
:7165
:7166      Error 1 - PWR FAIL INT caused interrupt before IPL was low enough to
:7167      allow it.
:7168      Error 2 - PWR FAIL INT failed to interrupt when IPL was at correct
:7169      value to allow it.
:7170      Error 3 - PWR FAIL INT produced incorrect identifier code. Should be 000(B)
:7171      Error 4 - INTRVL TIM INT caused interrupt before IPL was low enough to
:7172      allow it. Recieved data in printout is IPL at time of
:7173      interrupt, expected data is what IPL should have been before
:7174      interrupt was allowed.
:7175      Error 5 - INTRVL TIM INT failed to interrupt when IPL was at correct
:7176      value to allow it.
:7177      Error 6 - INTRVL TIM INT produced incorrect identifier code. Should be 001(B)
:7178      Error 7 - CONS ATTN caused interrupt before IPL was low enough to
:7179      allow it. Recieved data in printout is IPL at time of
:7180      interrupt, expected data is what IPL should have been before

```


ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

TEST 6 - INTR CTLR PAL, IPL, and BUS IRQ Test (M8390 CPO Module)
L 13
3-MAR-82 10:02:46
Fiche 1 Frame L13
Sequence 167
Page 161
Rev 01.00
ENKCB Micro-diagnostic for DAP module

:7181 : interrupt was allowed.
:7182 : Error 8 - CONS ATTN failed to interrupt when IPL was at correct
:7183 : value to allow it.
:7184 : Error 9 - CONS ATTN produced incorrect identifier code. Should be 101(B)
:7185 : Error A- BUS IRQ 7 caused interrupt before IPL was low enough to
:7186 : allow it. Recieved data in printout is IPL at time of
:7187 : interrupt, expected data is what IPL should have been before
:7188 : interrupt was allowed.
:7189 : Error B- BUS IRQ 7 failed to interrupt when IPL was at correct
:7190 : value to allow it.
:7191 : Error C- BUS IRQ 7 produced incorrect identifier code. Should be 010(B)
:7192 : Error D- BUS IRQ 6 caused interrupt before IPL was low enough to
:7193 : allow it. Recieved data in printout is IPL at time of
:7194 : interrupt, expected data is what IPL should have been before
:7195 : interrupt was allowed.
:7196 : Error E- BUS IRQ 6 failed to interrupt when IPL was at correct
:7197 : value to allow it.
:7198 : Error F- BUS IRQ 6 produced incorrect identifier code. Should be 011(B)
:7199 : Error 10- BUS IRQ 5 caused interrupt before IPL was low enough to
:7200 : allow it. Recieved data in printout is IPL at time of
:7201 : interrupt, expected data is what IPL should have been before
:7202 : interrupt was allowed.
:7203 : Error 11- BUS IRQ 5 failed to interrupt when IPL was at correct
:7204 : value to allow it.
:7205 : Error 12- BUS IRQ 5 produced incorrect identifier code. Should be 100(B)
:7206 : Error 13- BUS IRQ 4 caused interrupt before IPL was low enough to
:7207 : allow it. Recieved data in printout is IPL at time of
:7208 : interrupt, expected data is what IPL should have been before
:7209 : interrupt was allowed.
:7210 : Error 14- BUS IRQ 4 failed to interrupt when IPL was at correct
:7211 : value to allow it.
:7212 : Error 15- BUS IRQ 4 produced incorrect identifier code. Should be 110(B)
:7213 :
:7214 :
:7215 :
:7216 :
:7217 :
:7218 :
:7219 :
:7220 :
:7221 :
:7222 :
:7223 :
:7224 :
:7225 :
:7226 :
:7227 :
:7228 :
:7229 :
:7230 :
:7231 :
:7232 :
:7233 :
:7234 :
:7235 :

Debug:

NOTE: If there are any devices on the UNIBUS, it is possible that they could cause a BR interrupt to be asserted on the UNIBUS. Any devices that are on the UNIBUS should be removed if they are suspected of causing a failure in this test.

Error 1 - This error would occur if the INTR CTLR PAL outputs a low on IRQ OUT L (pin 19). Check the IPL inputs for a value of 1E(H) or 1F(H) during the NOP with SCTL=skip if no interrupt. If this is correct, the INTR CTLR PAL is probably bad. If the IPL value is incorrect, check the output at the 8 bit latch that stores the PSL bits (IPL's, current mode, and I-TRAP). If these are incorrect, check the inputs to this latch during the MOVE instruction that loads the IPL. The input on the CLK (pin 11) is generated by the REG CONTROL PAL output LOAD PSL L and the CLOCK REGS H signal.

Error 2 - This error occurs if the INTR CTLR PAL fails to output a low on IRQ OUT L (pin 19). Check the IPL inputs to this PAL as in error 1. If these are correct, check IRQ 2 through IRQ 0 (pins 9, 12, and 13 respectively) for a 7. If this is correct, the PAL is probably bad. If these inputs are incorrect, check the PRI ENC chip for a low

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

M 13
TEST 6 - INTR CTRL 3-MAR-82 Fiche 1 Frame M13
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKGB Micro-diagnostic for DAP module Sequence 168
TEST 6 - INTR CTRL PAL, IPL, and BUS IRQ Test (M8390 CPO Module) Rev 01.00 Page 162

:7236 : on the enable and D7. The PRI ENC chip is bad if these are correct.
:7237 : Otherwise the signal PWR FAIL INT L must be traced back.
:7238 :
:7239 : Error 3 - If this is the first error, the most likely suspect is the
:7240 : INTR CTRL PAL itself.
:7241 :
:7242 : Error 4 - See error 1. The only difference is that the IPL should be
:7243 : 18(H) or higher.
:7244 :
:7245 : Error 5 - See error 2. The only differences are that IPL should be 18
:7246 : (H) or higher and IRQ 2 through IRQ 0 should equal 6. INTRVL TIM INT
:7247 : is the asserted interrupt.
:7248 :
:7249 : Error 6 - Suspect the INTR CTRL PAL.
:7250 :
:7251 : Error 7 - See error 1. The only difference is that the IPL should be
:7252 : 14(H) or higher.
:7253 :
:7254 : Error 8 - See error 2. The only differences are that IPL should be 14
:7255 : (H) or higher and IRQ 2 through IRQ 0 should equal 2. CONS ATTN is
:7256 : the asserted interrupt.
:7257 :
:7258 : Error 9 - Suspect the INTR CTRL PAL.
:7259 :
:7260 : Error A - See error 1. The only difference is that the IPL should be
:7261 : 17(H) or higher.
:7262 :
:7263 : Error B - See error 2. The only differences are that IPL should be
:7264 : 17(H) or higher and IRQ 2 through IRQ 0 should equal 5. BR7 is the
:7265 : asserted interrupt.
:7266 :
:7267 : Error C- Suspect the INTR CTRL PAL.
:7268 :
:7269 : Error D- See error 1. The only difference is that the IPL should be
:7270 : 16(H) or higher.
:7271 :
:7272 : Error E - See error 2. The only differences are that IPL should be
:7273 : 16(H) or higher and IRQ 2 through IRQ 0 should equal 4. BR6 is the
:7274 : asserted interrupt.
:7275 :
:7276 : Error F- Suspect the INTR CTRL PAL.
:7277 :
:7278 : Error 10- See error 1. The only difference is that the IPL should be
:7279 : 15(H) or higher.
:7280 :
:7281 : Error 11 - See error 2. The only differences are that IPL should be
:7282 : 15(H) or higher and IRQ 2 through IRQ 0 should equal 3. BR5 is the
:7283 : asserted interrupt.
:7284 :
:7285 : Error 12- Suspect the INTR CTRL PAL.
:7286 :
:7287 : Error 13- See error 1. The only difference is that the IPL should be
:7288 : 14(H) or higher.
:7289 :
:7290 : Error 14 - See error 2. The only differences are that IPL should be

```

:7291      : 14(H) or higher and IRQ 2 through IRQ 0 should equal 1. BR4 is the
:7292      : asserted interrupt.
:7293
:7294      : Error 15- Suspect the INTR CTRL PAL.
:7295
:7296      :
:7297      :
:7298      :
:7299      :
:7300      :
:7301      :
:7302      :
:7303      :
:7304      :
:7305      :
:7306      :
:7307      :
:7308      :
:7309      :
:7310      :
:7311      :
:7312      :
:7313      :
:7314      :
:7315      :
:7316      :
:7317      :
:7318      :
:7319      :
:7320      :
:7321      :
:7322      :
:7323      :
:7324      :
:7325      :
:7326      :
:7327      :
:7328      :
:7329      :
:7330      :
:7331      :
:7332      :
:7333      :
:7334      :
:7335      :
:7336      :
:7337      :
:7338      :
:7339      :
:7340      :
:7341      :
:7342      :
:7343      :
:7344      :
:7345      :
:7346      :
:7347      :
:7348      :
:7349      :
:7350      :
:7351      :
:7352      :
:7353      :
:7354      :
:7355      :
:7356      :
:7357      :
:7358      :
:7359      :
:7360      :
:7361      :
:7362      :
:7363      :
:7364      :
:7365      :
:7366      :
:7367      :
:7368      :
:7369      :
:7370      :
:7371      :
:7372      :
:7373      :
:7374      :
:7375      :
:7376      :
:7377      :
:7378      :
:7379      :
:7380      :
:7381      :
:7382      :
:7383      :
:7384      :
:7385      :
:7386      :
:7387      :
:7388      :
:7389      :
:7390      :
:7391      :
:7392      :
:7393      :
:7394      :
:7395      :
:7396      :
:7397      :
:7398      :
:7399      :
:7400      :
:7401      :
:7402      :
:7403      :
:7404      :
:7405      :
:7406      :
:7407      :
:7408      :
:7409      :
:7410      :
:7411      :
:7412      :
:7413      :
:7414      :
:7415      :
:7416      :
:7417      :
:7418      :
:7419      :
:7420      :
:7421      :
:7422      :
:7423      :
:7424      :
:7425      :
:7426      :
:7427      :
:7428      :
:7429      :
:7430      :
:7431      :
:7432      :
:7433      :
:7434      :
:7435      :
:7436      :
:7437      :
:7438      :
:7439      :
:7440      :
:7441      :
:7442      :
:7443      :
:7444      :
:7445      :
:7446      :
:7447      :
:7448      :
:7449      :
:7450      :
:7451      :
:7452      :
:7453      :
:7454      :
:7455      :
:7456      :
:7457      :
:7458      :
:7459      :
:7460      :
:7461      :
:7462      :
:7463      :
:7464      :
:7465      :
:7466      :
:7467      :
:7468      :
:7469      :
:7470      :
:7471      :
:7472      :
:7473      :
:7474      :
:7475      :
:7476      :
:7477      :
:7478      :
:7479      :
:7480      :
:7481      :
:7482      :
:7483      :
:7484      :
:7485      :
:7486      :
:7487      :
:7488      :
:7489      :
:7490      :
:7491      :
:7492      :
:7493      :
:7494      :
:7495      :
:7496      :
:7497      :
:7498      :
:7499      :
:7500      :
:7501      :
:7502      :
:7503      :
:7504      :
:7505      :
:7506      :
:7507      :
:7508      :
:7509      :
:7510      :
:7511      :
:7512      :
:7513      :
:7514      :
:7515      :
:7516      :
:7517      :
:7518      :
:7519      :
:7520      :
:7521      :
:7522      :
:7523      :
:7524      :
:7525      :
:7526      :
:7527      :
:7528      :
:7529      :
:7530      :
:7531      :
:7532      :
:7533      :
:7534      :
:7535      :
:7536      :
:7537      :
:7538      :
:7539      :
:7540      :
:7541      :
:7542      :
:7543      :
:7544      :
:7545      :
:7546      :
:7547      :
:7548      :
:7549      :
:7550      :
:7551      :
:7552      :
:7553      :
:7554      :
:7555      :
:7556      :
:7557      :
:7558      :
:7559      :
:7560      :
:7561      :
:7562      :
:7563      :
:7564      :
:7565      :
:7566      :
:7567      :
:7568      :
:7569      :
:7570      :
:7571      :
:7572      :
:7573      :
:7574      :
:7575      :
:7576      :
:7577      :
:7578      :
:7579      :
:7580      :
:7581      :
:7582      :
:7583      :
:7584      :
:7585      :
:7586      :
:7587      :
:7588      :
:7589      :
:7590      :
:7591      :
:7592      :
:7593      :
:7594      :
:7595      :
:7596      :
:7597      :
:7598      :
:7599      :
:7600      :
:7601      :
:7602      :
:7603      :
:7604      :
:7605      :
:7606      :
:7607      :
:7608      :
:7609      :
:7610      :
:7611      :
:7612      :
:7613      :
:7614      :
:7615      :
:7616      :
:7617      :
:7618      :
:7619      :
:7620      :
:7621      :
:7622      :
:7623      :
:7624      :
:7625      :
:7626      :
:7627      :
:7628      :
:7629      :
:7630      :
:7631      :
:7632      :
:7633      :
:7634      :
:7635      :
:7636      :
:7637      :
:7638      :
:7639      :
:7640      :
:7641      :
:7642      :
:7643      :
:7644      :
:7645      :
:7646      :
:7647      :
:7648      :
:7649      :
:7650      :
:7651      :
:7652      :
:7653      :
:7654      :
:7655      :
:7656      :
:7657      :
:7658      :
:7659      :
:7660      :
:7661      :
:7662      :
:7663      :
:7664      :
:7665      :
:7666      :
:7667      :
:7668      :
:7669      :
:7670      :
:7671      :
:7672      :
:7673      :
:7674      :
:7675      :
:7676      :
:7677      :
:7678      :
:7679      :
:7680      :
:7681      :
:7682      :
:7683      :
:7684      :
:7685      :
:7686      :
:7687      :
:7688      :
:7689      :
:7690      :
:7691      :
:7692      :
:7693      :
:7694      :
:7695      :
:7696      :
:7697      :
:7698      :
:7699      :
:7700      :
:7701      :
:7702      :
:7703      :
:7704      :
:7705      :
:7706      :
:7707      :
:7708      :
:7709      :
:7710      :
:7711      :
:7712      :
:7713      :
:7714      :
:7715      :
:7716      :
:7717      :
:7718      :
:7719      :
:7720      :
:7721      :
:7722      :
:7723      :
:7724      :
:7725      :
:7726      :
:7727      :
:7728      :
:7729      :
:7730      :
:7731      :
:7732      :
:7733      :
:7734      :
:7735      :
:7736      :
:7737      :
:7738      :
:7739      :
:7740      :
:7741      :
:7742      :
:7743      :
:7744      :
:7745      :
:7746      :
:7747      :
:7748      :
:7749      :
:7750      :
:7751      :
:7752      :
:7753      :
:7754      :
:7755      :
:7756      :
:7757      :
:7758      :
:7759      :
:7760      :
:7761      :
:7762      :
:7763      :
:7764      :
:7765      :
:7766      :
:7767      :
:7768      :
:7769      :
:7770      :
:7771      :
:7772      :
:7773      :
:7774      :
:7775      :
:7776      :
:7777      :
:7778      :
:7779      :
:7780      :
:7781      :
:7782      :
:7783      :
:7784      :
:7785      :
:7786      :
:7787      :
:7788      :
:7789      :
:7790      :
:7791      :
:7792      :
:7793      :
:7794      :
:7795      :
:7796      :
:7797      :
:
```

```

U 167A, BE82.15 :7346      MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR NUMBER = 4
U 167B, 096C.EC :7347      JSR [SUB.T6]
U 167C, 8967.94 :7348      JMP [LOOP.T6.4] ;ERROR LOOP IF ENABLED
                          :7349
U 167D, 3660.15 :7350      MOV LS[INTERUPT.EN] TO WR[0]
U 167E, C768.15 :7351      BIS LS[CONSOLE.ATTN] TO WR[0]
U 167F, 3E80.15 :7352      MOV WR[0] TO LS[CONTROL.STATUS] ;SET UP CSR TO ASSERT CONS ATTN L
U 1680, 10E0.15 :7353      MISC [SET.CP.ATTN] ;GET CONSOLE'S ATTENTION
                          :7354
WAIT.T6.7:
U 1681, 0968.14 :7355      JMP [WAIT.T6.7] ;LOOP UNTIL CONSOLE RESPONDS
U 1682, 3611.15 :7356      MOV LS[T8] TO WR[2] ;WR2 = 1F 00 00
U 1683, B611.95 :7357      MOV LS[T8] TO WR[3]
U 1684, C567.95 :7358      BIC LS[BIT19] TO WR[3]
U 1685, 4565.95 :7359      BIC LS[BIT18] TO WR[3] ;WR3 = 13 00 00
U 1686, 3644.15 :7360      MOV LS[BIT2] TO WR[0]
U 1687, 4748.15 :7361      BIS LS[BIT4] TO WR[0]
U 1688, C74A.15 :7362      BIS LS[BIT5] TO WR[0]
U 1689, BE12.15 :7363      MOV WR[0] TO LS[T9] ;IDENTIFER CODE = 1101 00 (B)
                          :7364
LOOP.T6.7:
U 168A, B646.15 :7365      MOV LS[#8] TO WR[0]
U 168B, C140.15 :7366      SUB LS[#1] FROM WR[0]
U 168C, BE82.15 :7367      MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR NUMBER = 7
U 168D, 096C.EC :7368      JSR [SUB.T6]
U 168E, 8968.A4 :7369      JMP [LOOP.T6.7] ;LOOP ON ERROR IF ENABLED
                          :7370
U 168F, 3660.15 :7371      MOV LS[INTERUPT.EN] TO WR[0]
U 1690, 3E80.15 :7372      MOV WR[0] TO LS[CONTROL.STATUS] ;SET UP CSR TO DEASSERT INTERRUPTS
U 1691, 8870.FC :7373      JSR [ASSERT.OTHER] ;'OTHER' DATA FIELD WILL BE USED
U 1692, 10E0.15 :7374      MISC [SET.CP.ATTN] ;GET CONSOLE'S ATTENTION
                          :7375
WAIT.T6.A:
U 1693, 0969.34 :7376      JMP [WAIT.T6.A] ;LOOP UNTIL CONSOLE RESPONDS
U 1694, 3611.15 :7377      MOV LS[T8] TO WR[2] ;WR2 = 1F 00 00
U 1695, B611.95 :7378      MOV LS[T8] TO WR[3]
U 1696, C561.95 :7379      BIC LS[BIT16] TO WR[3]
U 1697, C567.95 :7380      BIC LS[BIT19] TO WR[3] ;WR3 = 16 00 00
U 1698, B646.15 :7381      MOV LS[BIT3] TO WR[0]
U 1699, C74A.15 :7382      BIS LS[BIT5] TO WR[0]
U 169A, BE12.15 :7383      MOV WR[0] TO LS[T9] ;IDENTIFER CODE = 1010 00 (B)
U 169B, B646.15 :7384      MOV LS[BIT3] TO WR[0]
U 169C, 3E16.15 :7385      MOV WR[0] TO LS[T11] ;BUS IRQ VALUE = 7
                          :7386
LOOP.T6.A:
U 169D, B646.15 :7387      MOV LS[#8] TO WR[0]
U 169E, 4742.15 :7388      BIS LS[#2] TO WR[0]
U 169F, BE82.15 :7389      MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR NUMBER = A
U 16A0, 896F.5C :7390      JSR [SUB2.T6]
U 16A1, 8969.D4 :7391      JMP [LOOP.T6.A] ;LOOP ON ERROR IF ENABLED
                          :7392
U 16A2, 3611.15 :7393      MOV LS[T8] TO WR[2] ;WR2 = 1F 00 00
U 16A3, B611.95 :7394      MOV LS[T8] TO WR[3]
U 16A4, 4563.95 :7395      BIC LS[BIT17] TO WR[3]
U 16A5, C567.95 :7396      BIC LS[BIT19] TO WR[3] ;WR3 = 15 00 00
U 16A6, B646.15 :7397      MOV LS[BIT3] TO WR[0]
U 16A7, 4744.15 :7398      BIS LS[BIT2] TO WR[0]
U 16A8, C74A.15 :7399      BIS LS[BIT5] TO WR[0]
U 16A9, BE12.15 :7400      MOV WR[0] TO LS[T9] ;IDENTIFER CODE = 1011 00 (B)
  
```

```

U 16AA, 3644,15 :7401      MOV LS[BIT2] TO WR[0]
U 16AB, 3E16,15 :7402      MOV WR[0] TO LS[T11]          ;BUS IRQ VALUE = 6
                                LOOP.T6.D:
U 16AC, B646,15 :7404      MOV LS[#8] TO WR[0]
U 16AD, 4744,15 :7405      BIS LS[#4] TO WR[0]
U 16AE, 2040,15 :7406      INC WR[0]
U 16AF, BE82,15 :7407      MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR NUMBER = D
U 16B0, 896F,5C :7408      JSR [SUB2.T6]
U 16B1, 096A,C4 :7409      JMP [LOOP.T6.D]          ;LOOP ON ERROR IF ENABLED
                                :7410
U 16B2, 3611,15 :7411      MOV LS[T8] TO WR[2]          ;WR2 = 1F 00 00
U 16B3, B669,95 :7412      MOV LS[#100000] TO WR[3]      ;PUT 10 00 00 IN WR
U 16B4, C765,95 :7413      BIS LS[#40000] TO WR[3]      ;WR3 = 14 00 00
U 16B5, 3648,15 :7414      MOV LS[BIT4] TO WR[0]
U 16B6, C74A,15 :7415      BIS LS[BIT5] TO WR[0]
U 16B7, BE12,15 :7416      MOV WR[0] TO LS[T9]          ;IDENTIFER CODE = 1100 00 (B)
U 16B8, 3642,15 :7417      MOV LS[BIT1] TO WR[0]
U 16B9, 3E16,15 :7418      MOV WR[0] TO LS[T11]          ;BUS IRQ VALUE = 5
                                :7419
                                LOOP.T6.10:
U 16BA, 3648,15 :7420      MOV LS[#10] TO WR[0]
U 16BB, BE82,15 :7421      MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR NUMBER = 10
U 16BC, 896F,5C :7422      JSR [SUB2.T6]
U 16BD, 896B,A4 :7423      JMP [LOOP.T6.10]          ;LOOP ON ERROR IF ENABLED
                                :7424
U 16BE, 3611,15 :7425      MOV LS[T8] TO WR[2]          ;WR2 = 1F 00 00
U 16BF, B611,95 :7426      MOV LS[T8] TO WR[3]
U 16C0, 4565,95 :7427      BIC LS[BIT18] TO WR[3]
U 16C1, C567,95 :7428      BIC LS[BIT19] TO WR[3]      ;WR3 = 13 00 00
U 16C2, 3648,15 :7429      MOV LS[BIT4] TO WR[0]
U 16C3, C746,15 :7430      BIS LS[BIT3] TO WR[0]
U 16C4, C74A,15 :7431      BIS LS[BIT5] TO WR[0]
U 16C5, BE12,15 :7432      MOV WR[0] TO LS[T9]          ;IDENTIFER CODE = 1110 00 (B)
U 16C6, B640,15 :7433      MOV LS[BIT0] TO WR[0]
U 16C7, 3E16,15 :7434      MOV WR[0] TO LS[T11]          ;BUS IRQ VALUE = 4
                                :7435
                                LOOP.T6.13:
U 16C8, 3648,15 :7436      MOV LS[#10] TO WR[0]
U 16C9, C0C0,15 :7437      ADD LS[#3(H)] TO WR[0]
U 16CA, BE82,15 :7438      MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR NUMBER = 13
U 16CB, 896F,5C :7439      JSR [SUB2.T6]
U 16CC, 896C,84 :7440      JMP [LOOP.T6.13]          ;LOOP ON ERROR IF ENABLED
U 16CD, 8972,14 :7441      JMP [END.T6]              ;ALL DONE
                                :7442
                                SUB.T6:
                                SUBT6.1:
U 16CE, 8870,FC :7444      JSR [ASSERT.OTHER]          ;"OTHER" DATA FIELD WILL BE USED
U 16CF, 0870,8C :7445      JSR [ASSERT.NA]
U 16D0, 3E89,15 :7446      MOV WR[2] TO LS[ADDRESS.DATA] ;WR2 WILL BE SHOWN IN "OTHER" DATA FIELD
U 16D1, 3FFF,15 :7447      MOV WR[2] TO LS[PSL.HW]      ;LOAD IPL WITH WR2
                                :7448
U 16D2, AB85,B5 :7449      XOR WR[2] WITH WR[3] TO 0,
U 16D3, 896D,D9 :7450      DT(LONG)&SET.ALU.CC          ;ARE WE DOWN TO IPL THAT SHOUL ALLOW INT?
U 16D4, DB00,07 :7451      JMP.IF[EQ] TO [SUBT6.2]      ;JUMP IF SO
U 16D5, 096D,74 :7452      SKIP.IF[NO.INTERRUPT]      ;CHECK FOR SKIP
U 16D6, 096D,B4 :7453      JMP [SUBT6.ERROR.1]          ;REPORT ERROR -INTERRUPT PRESENT-
                                :7454
                                SUBT6.ERROR.1:
U 16D7, 2F80,15 :7455      CLR WR[0]
  
```

U 16D8, 369F,95	:7456	MOV LS[ONES] TO WR[1]	:SET ERRONEOUS EXPECTED AND RECEIVED DATA TO FLAG ERROR
U 16D9, 0869,3C	:7457	JSR [CHECK.RESULT]	:REPORT ERROR
U 16DA, 5B00,14	:7458	RETURN	:LOOP ON ERROR IF ENABLED
	:7459	SUBT6.NOERROR.1:	
U 16DB, C161,15	:7460	SUB LS[#10000] FROM WR[2]	:WR2 = WR2 - 1 00 00 (NEXT LOWEST IPL)
U 16DC, 896C,E4	:7461	JMP [SUBT6.1]	:GO BACK AND TRY WITH NEXT IPL
	:7462	SUBT6.2:	
U 16DD, 8870,0C	:7463	JSR [INC.EN]	:INCREMENT ERROR NUMBER
U 16DE, 0870,8C	:7464	JSR [ASSERT.NA]	
U 16DF, DB00,07	:7465	SKIP.IF[NO.INTERRUPT]	:CHECK FOR NO SKIP
U 16E0, 896E,54	:7466	JMP [SUBT6.3]	:JUMP OVER ERROR REPORT
U 16E1, 2F80,15	:7467	CLR WR[0]	
U 16E2, 369E,95	:7468	MOV LS[ONES] TO WR[1]	:SET ERRONEOUS EXPECTED AND RECEIVED DATA TO FLAG ERROR
U 16E3, 0869,3C	:7469	JSR [CHECK.RESULT]	:REPORT ERROR
U 16E4, 5B00,14	:7470	RETURN	:LOOP ON ERROR IF ENABLED
	:7471	SUBT6.3:	
U 16E5, 8870,0C	:7472	JSR [INC.EN]	:INCREMENT ERROR NUMBER
U 16E6, 0870,8C	:7473	JSR [DEASSERT.NA]	
U 16E7, 3E89,15	:7474	MOV WR[2] TO LS[ADDRESS.DATA]	
U 16E8, 36FC,15	:7475	MOV LS[INTERRUPT.VEC] TO WR[0]	:GET IDENTIFIER CODE
U 16E9, B612,95	:7476	MOV LS[T9] TO WR[1]	:GET EXPECTED
U 16EA, 0869,3C	:7477	JSR [CHECK.RESULT]	:CHECK FOR PROPER IDENTIFIER CODE
U 16EB, 5B00,14	:7478	RETURN	:LOOP ON ERROR IF ENABLED
	:7479	TST WR[2]	
U 16EC, 2005,35	:7480	DT(LONG)&SET.ALU.CC	:WR2 = 0 ?
U 16ED, 096F,49	:7481	JMP.IF[EQL] TO [SUBT6.END]	:JUMP IF SO
U 16EE, C161,15	:7482	SUB LS[#10000] FROM WR[2]	:WR2 = WR2 - 1 00 00
U 16EF, 3E89,15	:7483	MOV WR[2] TO LS[ADDRESS.DATA]	:WR2 WILL BE SHOWN IN 'OTHER' DATA FIELD
U 16F0, 3FFF,15	:7484	MOV WR[2] TO LS[PSL.HW]	:LOAD PSL IPL BITS
U 16F1, 0870,4C	:7485	JSR [DEC.EN]	
U 16F2, 0870,4C	:7486	JSR [DEC.EN]	
U 16F3, 096D,D4	:7487	JMP [SUBT6.2]	:GO DO WITH LOWER IPL
	:7488	SUBT6.END:	
U 16F4, DB00,16	:7489	RETURN+1	
	:7490		
	:7491	SUB2.T6:	
	:7492	SUB2T6.1:	
U 16F5, 0870,8C	:7493	JSR [ASSERT.NA]	
U 16F6, 3E89,15	:7494	MOV WR[2] TO LS[ADDRESS.DATA]	:WR2 WILL BE SHOWN IN 'OTHER' DATA FIELD
U 16F7, 3FFF,15	:7495	MOV WR[2] TO LS[PSL.HW]	:LOAD IPL WITH WR2
U 16F8, B616,15	:7496	MOV LS[T11] TO WR[0]	
U 16F9, 3EF8,15	:7497	MOV WR[0] TO LS[OS]	
	:7498	XOR WR[2] WITH WR[3] TO 0,	
U 16FA, AB85,B5	:7499	DT(LONG)&SET.ALU.CC	:ARE WE DOWN TO IPL THAT SHOUL ALLOW INT?
U 16FB, 8970,79	:7500	JMP.IF[EQL] TO [SUB2T6.2]	:JUMP IF SO
U 16FC, 10F8,15	:7501	MISC2 [READ.ACC.UPC]	:LOAD OS TO BUS IRQ
U 16FD, DB00,15	:7502	NOP	
U 16FE, DB00,07	:7503	SKIP.IF[NO.INTERRUPT]	:CHECK FOR SKIP
U 16FF, 0970,14	:7504	JMP [SUB2T6.ERROR.1]	:REPORT ERROR -INTERRUPT PRESENT-
U 1700, 8970,54	:7505	JMP [SUB2T6.NOERROR.1]	
	:7506	SUB2T6.ERROR.1:	
U 1701, 2F80,15	:7507	CLR WR[0]	
U 1702, 369E,95	:7508	MOV LS[ONES] TO WR[1]	:SET ERRONEOUS EXPECTED AND RECEIVED DATA TO FLAG ERROR
U 1703, 0869,3C	:7509	JSR [CHECK.RESULT]	:REPORT ERROR
U 1704, 5B00,14	:7510	RETURN	:LOOP ON ERROR IF ENABLED

```

:7511 SUB2T6.NOERROR.1:
U 1705, C161,15 :7512 SUB LS[#10000] FROM WR[2] ;WR2 = WR2 - 1 00 00 (NEXT LOWEST IPL)
U 1706, 096F,54 :7513 JMP [SUB2T6.1] ;GO BACK AND TRY WITH NEXT IPL
:7514 SUB2T6.2:
U 1707, 8870,0C :7515 JSR [INC.EN] ;INCREMENT ERROR NUMBER
U 1708, 0870,8C :7516 JSR [ASSERT.NA]
U 1709, 10F8,15 :7517 MISC2 [READ.ACC.UPC] ;LOAD OS TO BUS IRQ
U 170A, DB00,15 :7518 NOP
U 170B, 36FD,95 :7519 MOV LS[INTERRUPT.VEC] TO WR[3] ;GET IDENTIFIER CODE
U 170C, DB00,07 :7520 SKIP,IF[NO.INTERRUPT] ;CHECK FOR NO SKIP
U 170D, 8971,24 :7521 JMP [SUB2T6.NOERROR.2] ;JUMP OVER ERROR REPORT
U 170E, 2F80,15 :7522 CLR WR[0]
U 170F, 369E,95 :7523 MOV LS[ONES] TO WR[1] ;SET ERRONEOUS EXPECTED AND RECEIVED DATA TO FLAG ERROR
U 1710, 0869,3C :7524 JSR [CHECK.RESULT] ;REPORT ERROR
U 1711, 5B00,14 :7525 RETURN ;LOOP ON ERROR IF ENABLED
:7526 SUB2T6.NOERROR.2:
U 1712, 8870,0C :7527 JSR [INC.EN] ;INCREMENT ERROR NUMBER
:7528 SUB2T6.3:
U 1713, 0870,8C :7529 JSR [DEASSERT.NA]
U 1714, A006,15 :7530 MOV WR[3] TO WR[0] ;GET IDENTIFIER CODE
U 1715, B612,95 :7531 MOV LS[T9] TO WR[1] ;GET EXPECTED
U 1716, 0869,3C :7532 JSR [CHECK.RESULT] ;CHECK FOR PROPER IDENTIFIER CODE
U 1717, 5B00,14 :7533 RETURN ;LOOP ON ERROR IF ENABLED
:7534 TST WR[2]
U 1718, 2005,35 :7535 DT(LONG)&SET.ALU.CC ;WR2 = 0 ?
U 1719, 8972,09 :7536 JMP,IF[EQL] TO [SUB2T6.END] ;JUMP IF SO
U 171A, C161,15 :7537 SUB LS[#10000] FROM WR[2] ;WR2 = WR2 - 1 00 00
U 171B, 3E89,15 :7538 MOV WR[2] TO LS[ADDRESS.DATA] ;WR2 WILL BE SHOWN IN 'OTHER' DATA FIELD
U 171C, 3FFF,15 :7539 MOV WR[2] TO LS[PSL.HW] ;LOAD PSL IPL BITS
U 171D, 0870,4C :7540 JSR [DEC.EN]
U 171E, 0870,4C :7541 JSR [DEC.EN]
U 171F, 0970,74 :7542 JMP [SUB2T6.2] ;GO DO WITH LOWER IPL
:7543 SUB2T6.END:
U 1720, DB00,16 :7544 RETURN+1
:7545 END.T6:
  
```

:7546 Page "TEST 7 - PRI ENC Chip Test (M8390 CPU Module)"
:7547 ++

:7548
:7549 Test Description:
:7550

:7551 This test checks the PRI ENC chip used to feed the INTR CTRL PAL. The
:7552 PRI ENC is tested by asserting all interrupt signals and checking that
:7553 the highest priority gets through by checking the identifier code.
:7554 Then the highest priority interrupt is disabled and the test repeated
:7555 until only the lowest priority interrupt signal remains. The IPL will
:7556 be set to 0 during this test. The lowest priority interrupt (BR4) will
:7557 not be tested separately, as it was already checked separately in the
:7558 previous test.
:7559

:7560 Assumptions:
:7561

:7562 It is assumed that the previous interrupt tests have run successfully.
:7563

:7564 Test Steps:
:7565

- :7566 1) Set up error mask, error number, and module number in LS (for
:7567 error printouts) and clear previous error number in LS.
:7568 2) Issue CPU ATTN with control word set up to assert PWR FAIL INT,
:7569 INTRVL TIM INT, and CONS ATTN.
:7570 3) Set up IPL to 0.
:7571 4) Load WR 1 with identifier code for PWR FAIL (000(B)).
:7572 5) Load OS 3-0 with 1's and execute MISC instruction to load BUS IRQ
:7573 7-4 with 1's.
:7574 6) Execute 2 NOP's to load IRQ latch and INTR CTRL PAL.
:7575 7) Execute MOVE with LS=7E to WR 0 to load the identifier code.
:7576 Check the result.
:7577 8) Issue CPU ATTN with control word to deassert PWR FAIL INT while
:7578 leaving other 2 interrupts on.
:7579 9) Load WR 1 with identifier code for INTRVL TIM INT (001(B)) and
:7580 repeat steps 5 through 7.
:7581 10) Issue CPU ATTN with control word to deassert INTRVL TIM INT while
:7582 leaving other interrupt on.
:7583 11) Load WR 1 with identifier code for BR7 (010(B)) and repeat steps
:7584 5 through 7.
:7585 12) Load WR 1 with identifier code for BR6 (011(B)).
:7586 13) Load OS 3-0 with 0111(B) and execute MISC instruction to load BUS
:7587 IRQ 7-4 with 0111(B). Repeat steps 6 and 7.
:7588 14) Load WR 1 with identifier code for BR5 (100(B)).
:7589 15) Load OS 3-0 with 0011(B) and execute MISC instruction to load BUS
:7590 IRQ 7-4 with 0011(B). Repeat steps 6 and 7.
:7591 16) Load WR 1 with identifier code for CONS ATTN (101(B)).
:7592 17) Load OS 3-0 with 0001(B) and execute MISC instruction to load BUS
:7593 IRQ 7-4 with 0001(B). Repeat steps 6 and 7.
:7594

:7595 Errors:
:7596

- :7597 Error 1 - PWR FAIL INT failed to override lower priority interrupts.
:7598 Error 2 - INTRVL TIM INT failed to override lower priority interrupts.
:7599 Error 3 - BR7 (BUS IRQ 7) failed to override lower priority interrupts.
:7600 Error 4 - BR6 (BUS IRQ 6) failed to override lower priority interrupts.


```

:7601      Error 5 - BR5 (BUS IRQ 5) failed to override lower priority interrupts.
:7602      Error 6 - CONS ATTN failed to override lower priority interrupt.
:7603
:7604      Debug:
:7605
:7606      NOTE: If there are any devices on the UNIBUS, it is possible that they
:7607      could cause a BR interrupt to be asserted on the UNIBUS. Any devices
:7608      that are on the UNIBUS should be removed if they are suspected of
:7609      causing a failure in this test.
:7610
:7611      Error 1 through 6 - For all these errors, the most likely suspect is
:7612      the PRI ENC chip itself. The output of the chip on pins 6, 7, and 8
:7613      should equal the binary value of the highest input value. For example
:7614      for error 1, the output should be 7, for error 2 it should be 6, etc.
:7615
:7616
:7617
:7618      T.7:
:7619      MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT15 IN WRO FOR CONTROL AND STATUS WORD
:7620      MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT15 IN CNTRL AND STATUS WORD
:7621      MISC [SET.CP.ATTN] ;BIT15 INDICATES START OF TEST TO CONSOLE
:7622      WAIT.T7.0: ;ISSUE CPU ATTN TO CONSOLE
:7623      JMP [WAIT.T7.0] ;LOOP HERE WAITING FOR CONSOLE TO RESPOND
:7624      MCOM LSLBIT2] TO WR[0]
:7625      BIC LS[BIT3] TO WR[0]
:7626      BIC LS[BIT4] TO WR[0]
:7627      BIC LS[BIT5] TO WR[0]
:7628      MOV WR[0] TO LS[ERROR.MASK] ;ERROR MASK = FFFFFFFD3
:7629      CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:7630      MOV LS[#1] TO WR[0] ;SET WR 0 TO 1
:7631      MOV WR[0] TO LS[ERROR.NUMBER] ;SET ERROR NUMBER TO FIRST ERROR
:7632      ROL WR[0] ;SET WRO TO 2
:7633      MOV WR[0] TO LS[MODULE.NUM] ;SET UP MODULE CODE FOR CPU MODULE
:7634      MOV LS[INTERUPT.EN] TO WR[0]
:7635      BIS LS[PWR.FAIL] TO WR[0] ;BIT TO ASSERT PWR FAIL INT
:7636      BIS LS[INTERVAL.TIM] TO WR[0] ;BIT TO ASSERT INTRVL TIM INT
:7637      BIS LS[CONSOLE.ATTN] TO WR[0] ;BIT TO ASSERT CONS ATTN
:7638      MOV WR[0] TO LS[CONTROL.STATUS]
:7639      MISC [SET.CP.ATTN]
:7640      WAIT.T7.1:
:7641      JMP [WAIT.T7.1] ;ASSERT INTERRUPTS
:7642      CLR WR[0] ;0 IN WR
:7643      MOV WR[0] TO LS[PSL.HW] ;SET IPL TO 0
:7644      LOOP.T7.1:
:7645      MOV LS[BIT5] TO WR[1] ;EXPECTED IDENT = 1000 00 (B)
:7646      MOV LS[#10] TO WR[0]
:7647      SUB LS[#1] FROM WR[0]
:7648      MOV WR[0] TO LS[OS] ;OS = #F
:7649      MISC2 [READ.ACC.UPC] ;LOAD BUS IRQ 7-4
:7650      NOP
:7651      MOV LS[INTERUPT.VEC] TO WR[0] ;GET IDENTIFIER CODE
:7652      JSR [CHECK.RESULT] ;CHECK IDENTIFIER CODE
:7653      JMP [LOOP.T7.1]
:7654      JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 2
:7655      MOV LS[INTERUPT.EN] TO WR[0]
  
```

U 1743.	4766.15	:7656	BIS LS[INTERVAL.TIM] TO WR[0]	:BIT TO ASSERT INTRVL TIM INT
U 1744.	C768.15	:7657	BIS LS[CONSOLE.ATTN] TO WR[0]	:BIT TO ASSERT CONS ATTN
U 1745.	3E80.15	:7658	MOV WR[0] TO LS[CONTROL.STATUS]	
U 1746.	10E0.15	:7659	MISC [SET.CP.ATTN]	
		:7660		
U 1747.	8974.74	:7661	WAIT.T7.2:	
		:7662	JMP [WAIT.T7.2]	:ASSERT INTERRUPTS
U 1748.	364A.95	:7663	LOOP.T7.2:	
U 1749.	C744.95	:7664	MOV LS[BIT5] TO WR[1]	
U 174A.	3648.15	:7665	BIS LS[BIT2] TO WR[1]	:EXPECTED IDENT = 1001 00 (B)
U 174B.	C140.15	:7666	MOV LS[#10] TO WR[0]	
U 174C.	3EF8.15	:7667	SUB LS[#1] FROM WR[0]	
U 174D.	10F8.15	:7668	MOV WR[0] TO LS[OS]	:OS = #F
U 174E.	DB00.15	:7669	MISC2 [READ.ACC.UPC]	:LOAD BUS IRQ 7-4
U 174F.	36FC.15	:7670	NOP	
U 1750.	0869.3C	:7671	MOV LS[INTERRUPT.VEC] TO WR[0]	:GET IDENTIFIER CODE
U 1751.	8974.84	:7672	JSR [CHECK.RESULT]	:CHECK IDENTIFIER CODE
U 1752.	8870.0C	:7673	JMP [LOOP.T7.2]	
U 1753.	3660.15	:7674	JSR [INC.EN]	:INCREMENT ERROR NUMBER TO 3
U 1754.	C768.15	:7675	MOV LS[INTERUPT.EN] TO WR[0]	
U 1755.	3E80.15	:7676	BIS LS[CONSOLE.ATTN] TO WR[0]	:BIT TO ASSERT CONS ATTN
U 1756.	10E0.15	:7677	MOV WR[0] TO LS[CONTROL.STATUS]	
		:7678	MISC [SET.CP.ATTN]	
U 1757.	0975.74	:7679	WAIT.T7.3:	
		:7680	JMP [WAIT.T7.3]	:ASSERT INTERRUPTS
U 1758.	3646.95	:7681	LOOP.T7.3:	
U 1759.	474A.95	:7682	MOV LS[BIT3] TO WR[1]	
U 175A.	3648.15	:7683	BIS LS[BIT5] TO WR[1]	:EXPECTED IDENT = 1010 00 (B)
U 175B.	C140.15	:7684	MOV LS[#10] TO WR[0]	
U 175C.	3EF8.15	:7685	SUB LS[#1] FROM WR[0]	
U 175D.	10F8.15	:7686	MOV WR[0] TO LS[OS]	:OS = #F
U 175E.	DB00.15	:7687	MISC2 [READ.ACC.UPC]	:LOAD BUS IRQ 7-4
U 175F.	36FC.15	:7688	NOP	
U 1760.	0869.3C	:7689	MOV LS[INTERRUPT.VEC] TO WR[0]	:GET IDENTIFIER CODE
U 1761.	0975.84	:7690	JSR [CHECK.RESULT]	:CHECK IDENTIFIER CODE
U 1762.	8870.0C	:7691	JMP [LOOP.T7.3]	
		:7692	JSR [INC.EN]	:INCREMENT ERROR NUMBER TO 4
U 1763.	3646.95	:7693	LOOP.T7.4:	
U 1764.	474A.95	:7694	MOV LS[BIT3] TO WR[1]	
U 1765.	C744.95	:7695	BIS LS[BIT5] TO WR[1]	
U 1766.	B646.15	:7696	BIS LS[BIT2] TO WR[1]	:EXPECTED IDENT = 1011 00 (B)
U 1767.	C140.15	:7697	MOV LS[#8] TO WR[0]	
U 1768.	3EF8.15	:7698	SUB LS[#1] FROM WR[0]	
U 1769.	10F8.15	:7699	MOV WR[0] TO LS[OS]	:OS = #7
U 176A.	DB00.15	:7700	MISC2 [READ.ACC.UPC]	:LOAD BUS IRQ 7-4
U 176B.	36FC.15	:7701	NOP	
U 176C.	0869.3C	:7702	MOV LS[INTERRUPT.VEC] TO WR[0]	:GET IDENTIFIER CODE
U 176D.	8976.34	:7703	JSR [CHECK.RESULT]	:CHECK IDENTIFIER CODE
U 176E.	8870.0C	:7704	JMP [LOOP.T7.4]	
		:7705	JSR [INC.EN]	:INCREMENT ERROR NUMBER TO 5
U 176F.	B648.95	:7706	LOOP.T7.5:	
U 1770.	474A.95	:7707	MOV LS[BIT4] TO WR[1]	
U 1771.	36C0.15	:7708	BIS LS[BIT5] TO WR[1]	:EXPECTED IDENT = 1100 00 (B)
U 1772.	3EF8.15	:7709	MOV LS[#3(H)] TO WR[0]	
U 1773.	10F8.15	:7710	MOV WR[0] TO LS[OS]	:OS = #3
			MISC2 [READ.ACC.UPC]	:LOAD BUS IRQ 7-4

```

U 1774, DB00,15 :7711      NOP
U 1775, 36FC,15 :7712      MOV LS[INTERRUPT.VEC] TO WR[0] ;GET IDENTIFIER CODE
U 1776, 0869,3C :7713      JSR [CHECK.RESULT] ;CHECK IDENTIFIER CODE
U 1777, 8976,F4 :7714      JMP [LOOP.T7.5]
U 1778, 8870,0C :7715      JSR [INC.EN] ;INCREMENT ERRCR NUMBER TO 6
                        :7716
L 1779, B648,95 :7717      LOOP.T7.6: MOV LS[BIT4] TO WR[1]
U 177A, 474A,95 :7718      BIS LS[BIT5] TO WR[1]
U 177B, C744,95 :7719      BIS LS[BIT2] TO WR[1] ;EXPECTED IDENT = 1101 00 (B)
U 177C, B640,15 :7720      MOV LS[#1] TO WR[0]
U 177D, 3EF8,15 :7721      MOV WR[0] TO LS[OS] ;OS = #1
U 177E, 10F8,15 :7722      MISC2 [READ.ACC.UPC] ;LOAD BUS IRQ 7-4
U 177F, DB00,15 :7723      NOP
U 1780, 36FC,15 :7724      MOV LS[INTERRUPT.VEC] TO WR[0] ;GET IDENTIFIER CODE
U 1781, 0869,3C :7725      JSR [CHECK.RESULT] ;CHECK IDENTIFIER CODE
U 1782, 0977,94 :7726      JMP [LOOP.T7.6]
U 1783, 3660,15 :7727      MOV LS[INTERUPT.EN] TO WR[0]
U 1784, 3E80,15 :7728      MOV WR[0] TO LS[CONTROL.STATUS] ;CSR = BIT16
U 1785, 10E0,15 :7729      MISC [SET.CP.ATTN] ;CALL CONSOLE
                        :7730
U 1786, 0978,64 :7731      WAIT.T7.CLRINT: JMP [WAIT.T7.CLRINT] ;CLEAR ALL INTERRUPTS
                        :7732
END.T7:
  
```

:7733 Page "TEST 8 - T-TRAP and Mask Test (M8390 CPU Module)"
:7734 ++

:7735
:7736 Test Description:
:7737

:7738 The T-TRAP test consists of enabling the T-TRAP interrupt and checking
:7739 that the correct identifier code is produced with the IPL at all ones
:7740 as long as no other interrupts are pending. The skip if no interrupt
:7741 is also checked. The T-TRAP bit is produced by setting either bit 4
:7742 or bit 30 and loading the PSL latch (the latch also outputs the IPL
:7743 bits and the current mode bits). All four combinations of bits 30 and
:7744 4 are checked.
:7745 Then with the IPL at 0, the interrupts BR4, CONS ATTN, CONS HALT
:7746 and BR6 are asserted one at a time, and the identifier code is checked
:7747 to assure that these interrupts over-ride the T-TRAP interrupt. The
:7748 MASK test consists of setting the IPL to 1F(H) and enabling both the
:7749 T-TRAP signal and CONS HALT (the IPL does not block either of these
:7750 interrupts). Then a MISC instruction is executed which sets the MASK
:7751 HALT AND T-TRAP signal from pin 9 of the decoder which is the input to
:7752 pin 18 of the INTR CTLR PAL. A NOP with skip if no interrupt is
:7753 executed. The instruction should skip since the interrupt is blocked
:7754 by the mask for this cycle. Then another NOP with skip if no interrupt
:7755 is executed. This time no skip should occur since the mask is only
:7756 valid for one cycle.
:7757

:7758 Assumptions:
:7759

:7760 It is assumed that all previous interrupt tests have run successfully.
:7761

:7762 Test Steps:
:7763

- :7764 1) Set up error mask, error number, and module number in LS (for
:7765 error printouts) and clear previous error number in LS.
- :7766 2) Load WR 0 with 40 1F 00 10(H) and execute a MOVE from WR 0 to LS
:7767 FF. This loads the IPL with 14 and also should set T-TRAP (Y BUS
:7768 bits 30 and 4 are set).
- :7769 3) Load WR 1 with 1111(B) in bits 5 through 2 (identifier code).
- :7770 4) Execute MOVE to WR 0 from LS 7E(H) (stat reg) and check identifier
:7771 code.
- :7772 5) Execute NOP with skip if no interrupt and check for no skip.
- :7773 6) Load WR 0 with 00 1F 00 10(H) and execute a MOVE from WR 0 to LS
:7774 FF. This loads the IPL with 14 and also should set T-TRAP (Y BUS
:7775 bit 4 is set).
- :7776 7) Load WR 1 with 1111(B) in bits 5 through 2 (identifier code) and
:7777 repeat step 4.
- :7778 8) Load WR 0 with 40 1F 00 00(H) and execute a MOVE from WR 0 to LS
:7779 FF. This loads the IPL with 14 and also should set T-TRAP (Y BUS
:7780 bit 30 is set).
- :7781 9) Load WR 1 with 1111(B) in bits 5 through 2 (identifier code) and
:7782 repeat step 4.
- :7783 10) Load WR 0 with 00 1F 00 00(H) and execute a MOVE from WR 0 to LS
:7784 FF. This loads the IPL with 14 but should not set T-TRAP (Y BUS
:7785 bits 30 and 4 are both clear).
- :7786 11) Load WR 1 with 1111(B) in bits 5 through 2 (identifier code for no
:7787 interrupt) and repeat step 4.

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

K 14
TEST 8 - T-TRAP and 3-MAR-82
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module
TEST 8 - T-TRAP and Mask Test (M8390 CPU Module)

Fiche 1 Frame K14
Sequence 179
Rev 01.00 Page 173

:7788 : 12) Load IPL with 0 and enable T-TRAP. Then assert BR4 via the OS reg.
:7789 : 13) Load WR 1 with 1110 (identifier code for BR4) and repeat step 4.
:7790 : 14) Assert BR6, load WR 1 with 0111 (identifier code for BR6) and re-
:7791 : peat step 4.
:7792 : 15) Execute CPU ATTN with bits 16 and 20 of the control and status
:7793 : word set to enable CONS ATTN.
:7794 : 16) Load WR 1 with 1101 (identifier code for CONS ATTN), execute a NOP
:7795 : and repeat step 4.
:7796 : 17) Execute CPU ATTN with bits 16 and 17 of the control and status
:7797 : word set to enable CONS HALT.
:7798 : 18) Load WR 1 with 0111 (identifier code for CONS HALT), execute a NOP
:7799 : and repeat step 4.
:7800 : 19) Execute CPU ATTN with bits 16 and 17 of the control and status
:7801 : word set to enable CONS HALT.
:7802 : 20) Load WR 1 with all ones to force an error if check result is call-
:7803 : ed. Load WR 0 with 40 1F 00 00(B) and execute a MOVE from WR 0 to
:7804 : LS FF. This loads the IPL with 14 and also sets T-TRAP (Y BUS bit
:7805 : 30 is set).
:7806 : 21) Execute MISC instruction to mask T-TRAP and halt interrupts.
:7807 : 22) Execute a NOP with skip if no interrupt. The in struction should
:7808 : skip.
:7809 : 23) Execute another NOP with skip if no interrupt. This time the in-
:7810 : struction should not skip.
:7811 : 24) Execute a BIS LS[6] TO WR[0] instruction. This sets the same bits
:7812 : at the decoder as MISC except MISC INSTR H should be low.
:7813 : 25) Execute another NOP with skip if no interrupt. Again the in-
:7814 : struction should not skip.

:7815 : Errors:

:7816 : Error 1 - T-TRAP did not produce correct identifier code with Y BUS
:7817 : bits 30 and 4 both set.
:7818 : Error 2 - T-TRAP did not produce an interrupt.
:7819 : Error 3 - T-TRAP did not produce correct identifier code with Y BUS
:7820 : bit 4 set.
:7821 : Error 4 - T-TRAP did not produce correct identifier code with Y BUS
:7822 : bit 30 set.
:7823 : Error 5 - T-TRAP did not get disabled with Y BUS bits 30 and 4 both
:7824 : clear.
:7825 : Error 6 -
:7826 : Error 7 - T-TRAP was not overridden by BR4.
:7827 : Error 8 - T-TRAP was not overridden by BR6.
:7828 : Error 9 - T-TRAP was not overridden by CONS ATTN
:7829 : Error A - T-TRAP was not overridden by CONS HALT
:7830 : Error B - MASK HALT AND T-BIT interrupt failed to mask interrupt.
:7831 : Error C - MASK HALT AND T-BIT interrupt failed to go away.
:7832 : Error D - MASK HALT AND T-BIT came up on a non-MISC instruction.

:7833 : Debug:

:7834 : Error 1 - This error indicates that either the T-TRAP did not get
:7835 : asserted or the INTR CTRL PAL is faulty. First check the signal T-
:7836 : TRAP H going into the INTR CTRL PAL during the MOVE instruction that
:7837 : loads the identifier code into WR 0. The signal should be high. If
:7838 : this input is OK, the PAL is probably bad. If this signal is low,
:7839 :
:7840 :
:7841 :
:7842 :

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

TEST 8 - T-TRAP and Mask Test (M8390 CPU Module)
MICRO2 1L(03) 3-MAR-82 10:02:46
TEST 8 - T-TRAP and Mask Test (M8390 CPU Module)

Fiche 1 Frame L14

Sequence 180
Rev 01.00 Page 174

:7843 : check the output directly out of the PSL latch that supplies this
:7844 : signal. If the signal is bad there, check the input T.OR.TP H during
:7845 : the instruction which loads the latch (MOVE to LS FF). It should be
:7846 : high during this instruction. If it is, the latch is probably bad.
:7847 : If T.OR.TP H is low, check the SHIFT DATA PAL for a high on BUS Y D30
:7848 : H and BUS Y D04 H. If either is high, the output T.OR.TP H should
:7849 : be high. If not, suspect the SHIFT DATA PAL.
:7850 :
:7851 : Error 2 - Suspect the INTR CTLR PAL.
:7852 :
:7853 : Error 3 - Suspect the SHIFT DATA PAL or the input BUS Y D04 H. This
:7854 : input should be high during the instruction that loads the PSL latch.
:7855 : If the input is OK, the SHIFT DATA PAL is probably bad.
:7856 :
:7857 : Error 4 - Suspect the SHIFT DATA PAL or the input BUS Y D30 H. This
:7858 : input should be high during the instruction that loads the PSL latch.
:7859 : If the input is OK, the SHIFT DATA PAL is probably bad.
:7860 :
:7861 : Error 5 - Suspect the SHIFT DATA PAL or the inputs BUS Y D04 H or BUS
:7862 : Y D30. These inputs should both be low during the instruction that
:7863 : loads the PSL latch. If the input is OK, the SHIFT DATA PAL is prob-
:7864 : ably bad.
:7865 :
:7866 : Error 6 -
:7867 :
:7868 : Error 7 - Suspect the INTR CTLR PAL.
:7869 :
:7870 : Error 8 - Suspect the INTR CTLR PAL.
:7871 :
:7872 : Error 9 - Suspect the INTR CTLR PAL.
:7873 :
:7874 : Error A - Suspect the INTR CTLR PAL.
:7875 :
:7876 : Error B - Check the MASK L input to the INTR CTLR PAL during the MISC
:7877 : instruction that sets the mask for a low. If this is correct, suspect
:7878 : the INTR CTLR PAL itself. Otherwise suspect the decoder which outputs
:7879 : MASK L.
:7880 :
:7881 : Error C - Either the signal MASK L is staying low even when a MISC
:7882 : is not executed, or there is a timing problem.
:7883 :
:7884 : Error D - Check the signal MISC INST H at the decoder chip that outputs
:7885 : MASK T-BIT AND HALT. If it is low during the BIS LS[6] TO WR[0] in-
:7886 : struction, then the decoder is probably bad.
:7887 :
:7888 :
:7889 :
:7890 :
:7891 :
:7892 :
:7893 :
:7894 :
:7895 :
:7896 :
:7897 :

T.8:--

U 1787, B65E,15
U 1788, 3E80,15
U 1789, 10E0,15
U 178A, 0978,A4
U 178B, 3660,15
U 178C, 3E80,15

MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT15 IN WRO FOR CONTROL AND STATUS WORD
MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT15 IN CNTRL AND STATUS WORD
MISC [SET.CP.ATTN] ;BIT15 INDICATES START OF TEST TO CONSOLE
ISSUE CPU ATTN TO CONSOLE
WAIT.T8.0:
JMP [WAIT.T8.0] ;LOOP HERE WAITING FOR CONSOLE TO RESPOND
MOV LS[BIT16] TO WR[0] ;CLEAR ALL INTERRUPTS
MOV WR[0] TO LS[CONTROL.STATUS]

M 14
 ZZ-ENKCB-1.0 : ENKCB.MIC TEST 8 - T-TRAP and 3-MAR-82 Fiche 1 Frame M14 Sequence 181
 : ENKCB.MCR MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Rev 01.00 Page 175
 : ENKCB.MIC TEST 8 - T-TRAP and Mask Test (M8390 CPU Module)

```

U 178D, 10E0,15 :7898 MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:7899 WAIT.T8.1: ;LOOP HERE WAITING FOR CONSOLE TO RESPOND
U 178E, 8978,E4 :7900 JMP [WAIT.T8.1]
U 178F, 5F44,15 :7901 MCOM LS[BIT2] TO WR[0]
U 1790, 4546,15 :7902 BIC LS[BIT3] TO WR[0]
U 1791, C548,15 :7903 BIC LS[BIT4] TO WR[0]
U 1792, 454A,15 :7904 BIC LS[BIT5] TO WR[0]
U 1793, 3E8A,15 :7905 MOV WR[0] TO LS[ERROR.MASK] ;ERROR MASK = FFFFFFF3
U 1794, 65A0,15 :7906 CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
U 1795, B640,15 :7907 MOV LS[#1] TO WR[0] ;SET WR 0 TO 1
U 1796, BE82,15 :7908 MOV WR[0] TO LS[ERROR.NUMBER] ;SET ERROR NUMBER TO FIRST ERROR
U 1797, A3C0,15 :7909 ROL WR[0] ;SET WRO TO 2
U 1798, 3E8C,15 :7910 MOV WR[0] TO LS[MODULE.NUM] ;SET UP MODULE CODE FOR CPU MODULE
:7911 LOOP.T8.1:
U 1799, B724,15 :7912 MOV LS[HI.IPL] TO WR[0]
U 179A, C77C,15 :7913 BIS LS[BIT30] TO WR[0]
U 179B, 4748,15 :7914 BIS LS[BIT4] TO WR[0] ;WRO = 40 1F 00 10
U 179C, BFFE,15 :7915 MOV WR[0] TO LS[PSL.HW] ;LOAD IPL AND SET T-TRAP
U 179D, 369E,95 :7916 MOV LS[ONES] TO WR[1] ;EXPECTED DATA = 1111(8) IN BITS 2 - 5
U 179E, 36FC,15 :7917 MOV LS[INTERRUPT.VEC] TO WR[0] ;GET IDENTIFIER CODE
U 179F, 0869,3C :7918 JSR [CHECK.RESULT] ;CHECK IDENTIFIER CODE
U 17A0, 8979,94 :7919 JMP [LOOP.T8.1] ;LOOP ON ERROR IF ENABLED
U 17A1, 8870,0C :7920 JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 2
U 17A2, 0870,8C :7921 JSR [ASSERT.NA] ;SET N/A CODE FOR EX & REC DATA FIELDS
:7922 LOOP.T8.2:
U 17A3, DB00,07 :7923 SKIP.IF[NO.INTERRUPT] ;SHOULD NOT SKIP
U 17A4, 897A,94 :7924 JMP [T8.OK.2] ;JUMP OVER ERROR CALL
U 17A5, 2F80,15 :7925 CLR WR[0]
U 17A6, 369E,95 :7926 MOV LS[ONES] TO WR[1] ;SET BAD DATA
U 17A7, 0869,3C :7927 JSR [CHECK.RESULT] ;REPORT ERROR
U 17A8, 897A,34 :7928 JMP [LOOP.T8.2] ;LOOP ON ERROR IF ENABLED
:7929 T8.OK.2:
U 17A9, 8870,0C :7930 JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 3
:7931 LOOP.T8.3:
U 17AA, B724,15 :7932 MOV LS[HI.IPL] TO WR[0]
U 17AB, 4748,15 :7933 BIS LS[BIT4] TO WR[0] ;WRO = 00 1F 00 10
U 17AC, BFFE,15 :7934 MOV WR[0] TO LS[PSL.HW] ;LOAD IPL AND SET T-TRAP
U 17AD, DB00,07 :7935 SKIP.IF[NO.INTERRUPT] ;SHOULD NOT SKIP
U 17AE, 097B,34 :7936 JMP [T8.OK.3] ;JUMP OVER ERROR CALL
U 17AF, 2F80,15 :7937 CLR WR[0]
U 17B0, 369E,95 :7938 MOV LS[ONES] TO WR[1] ;SET BAD DATA
U 17B1, 0869,3C :7939 JSR [CHECK.RESULT] ;REPORT ERROR
U 17B2, 897A,A4 :7940 JMP [LOOP.T8.3] ;LOOP ON ERROR IF ENABLED
:7941 T8.OK.3:
U 17B3, 8870,0C :7942 JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 4
:7943 LOOP.T8.4:
U 17B4, B724,15 :7944 MOV LS[HI.IPL] TO WR[0]
U 17B5, C77C,15 :7945 BIS LS[BIT30] TO WR[0] ;WRO = 40 1F 00 00
U 17B6, BFFE,15 :7946 MOV WR[0] TO LS[PSL.HW] ;LOAD IPL AND SET T-TRAP
U 17B7, DB00,07 :7947 SKIP.IF[NO.INTERRUPT] ;SHOULD NOT SKIP
U 17B8, 897B,D4 :7948 JMP [T8.OK.4] ;JUMP OVER ERROR CALL
U 17B9, 2F80,15 :7949 CLR WR[0]
U 17BA, 369E,95 :7950 MOV LS[ONES] TO WR[1] ;SET BAD DATA
U 17BB, 0869,3C :7951 JSR [CHECK.RESULT] ;REPORT ERROR
U 17BC, 897B,44 :7952 JMP [LOOP.T8.4] ;LOOP ON ERROR IF ENABLED
  
```

```

U 17BD, 8870.0C :7953 T8.OK.4: JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 5
:7954
U 17BE, 8724.15 :7955 LOOP.T8.5: MOV LS[HI.IPL] TO WR[0] ;WRO = 00 1F 00 00
:7956 MOV WR[0] TO LS[PSL.HW] ;LOAD IPL
U 17BF, 8FFE.15 :7957 SKIP.IF[NO.INTERRUPT] ;SHOULD SKIP
:7958 SKIP
U 17C0, DB00.07 :7959 JMP [T8.OK.5] ;JUMP OVER ERROR CALL
U 17C1, 5B00.1E :7960 CLR WR[0]
U 17C2, 097C.74 :7961 MOV LS[ONES] TO WR[1] ;SET BAD DATA
U 17C3, 2F80.15 :7962 JSR [CHECK.RESULT] ;REPORT ERROR
U 17C4, 369E.95 :7963 JMP [LOOP.T8.5] ;LOOP ON ERROR IF ENABLED
U 17C5, 0869.3C :7964
U 17C6, 897B.E4 :7965 T8.OK.5: JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 6
:7966
U 17C7, 8870.0C :7967 LOOP.T8.6: CLR WR[0]
:7968 BIS LS[BIT30] TO WR[0]
U 17C8, 2F80.15 :7969 BIS LS[BIT4] TO WR[0] ;WRO = 40 00 00 10
U 17C9, C77C.15 :7970 MOV WR[0] TO LS[PSL.HW] ;ENABLE T-TRAP
U 17CA, 4748.15 :7971 MOV LS[HI.IPL] TO WR[0]
U 17CB, 8FFE.15 :7972 BIS LS[BIT4] TO WR[0]
U 17CC, B724.15 :7973 MOV LS[PSL.HW] TO WR[0] ;READ RATHER THAN LOAD IPL AND SET T-TRAP
U 17CD, 4748.15 :7974 SKIP.IF[NO.INTERRUPT] ;SHOULD NOT SKIP
U 17CE, 37FE.15 :7975 JMP [T8.OK.6] ;INTERRUPT, END OF TEST
U 17CF, DB00.07 :7976 CLR WR[0] ;NO INTERRUPT OCCURRED
U 17D0, 097D.54 :7977 MOV LS[ONES] TO WR[1] ;SET UP BAD DATA
U 17D1, 2F80.15 :7978 JSR [CHECK.RESULT] ;REPORT ERROR
U 17D2, 369E.95 :7979 JMP [LOOP.T8.6] ;LOOP ON ERROR IF ENABLED
U 17D3, 0869.3C :7980
U 17D4, 097C.84 :7981 T8.OK.6: JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 7
:7982 JSR [DEASSERT.NA] ;RESTORE EX & REC DATA FIELDS
U 17D5, 8870.0C :7983
U 17D6, 0870.BC :7984 LOOP.T8.7: MOV LS[BIT4] TO WR[0]
:7985 MOV WR[0] TO LS[PSL.HW] ;CLEAR IPL AND ENABLE T-TRAP
U 17D7, 3648.15 :7986 MOV LS[BIT0] TO WR[0]
U 17D8, 8FFE.15 :7987 MOV WR[0] TO LS[OS] ;OS = BIT 0
U 17D9, B640.15 :7988 MISC2 [READ.ACC.UPC] ;ASSERT BR4
U 17DA, 3EF8.15 :7989 NOP
U 17DB, 10F8.15 :7990 MOV LS[INTERRUPT.VEC] TO WR[0] ;GET IDENTIFIER CODE
U 17DC, DB00.15 :7991 MOV LS[BIT5] TO WR[1]
U 17DD, 36FC.15 :7992 BIS LS[BIT4] TO WR[1]
U 17DE, 364A.95 :7993 BIS LS[BIT3] TO WR[1] ;EXPECTED DATA = 1110(B) IN BITS 2-5
U 17DF, C748.95 :7994 JSR [CHECK.RESULT] ;CHECK IDENTIFIER CODE
U 17E0, 4746.95 :7995 JMP [LOOP.T8.7] ;LOOP ON ERROR IF ENABLED
U 17E1, 0869.3C :7996 JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 8
U 17E2, 897D.74 :7997
U 17E3, 8870.0C :7998 LOOP.T8.8: MOV LS[BIT4] TO WR[0]
:7999 MOV WR[0] TO LS[PSL.HW] ;CLEAR IPL AND ENABLE T-TRAP
U 17E4, 3648.15 :8000 MOV LS[BIT2] TO WR[0]
U 17E5, 8FFE.15 :8001 MOV WR[0] TO LS[OS] ;OS = 6
U 17E6, 3644.15 :8002 MISC2 [READ.ACC.UPC] ;ASSERT BR6
U 17E7, 3EF8.15 :8003 NOP
U 17E8, 10F8.15 :8004 MOV LS[INTERRUPT.VEC] TO WR[0] ;GET IDENTIFIER CODE
U 17E9, DB00.15 :8005 MOV LS[BIT5] TO WR[1]
U 17EA, 36FC.15 :8006 BIS LS[BIT3] TO WR[1]
U 17EB, 364A.95 :8007
U 17EC, 4746.95 :8007
  
```



```

U 17ED, C744,95 :8008      BIS LS[BIT2] TO WR[1]      ;EXPECTED DATA = 1011(B) IN BITS 2-5
U 17EE, 0869,3C :8009      JSR [CHECK.RESULT]      ;CHECK IDENTIFIER CODE
U 17EF, 897E,44 :8010      JMP [LOOP.T8.8]      ;LOOP ON ERROR IF ENABLED
U 17F0, 8870,0C :8011      JSR [INC.EN]      ;INCREMENT ERROR NUMBER TO 9
:8012
U 17F1, 3660,15 :8013      LOOP.T8.9: MOV LS[INTERUPT.EN] TO WR[0]
U 17F2, C768,15 :8014      BIS LS[CONSOLE.ATTN] TO WR[0]
U 17F3, 3E80,15 :8015      MOV WR[0] TO LS[CONTROL.STATUS] ;CSR = BIT16:BIT20
U 17F4, 10E0,15 :8016      MISC [SET.CP.ATTN] ;CALL CONSOLE
:8017
U 17F5, 897F,54 :8018      WAIT.T8.9: JMP [WAIT.T8.9] ;ASSERT CONS ATTN
U 17F6, 364A,95 :8019      MOV LS[BIT5] TO WR[1]
U 17F7, C748,95 :8020      BIS LS[BIT4] TO WR[1]
U 17F8, C744,95 :8021      BIS LS[BIT2] TO WR[1] ;EXPECTED DATA = 1101(B) IN BITS 2-5
U 17F9, 36FC,15 :8022      MOV LS[INTERUPT.VEC] TO WR[0] ;GET IDENTIFIER CODE
U 17FA, 0869,3C :8023      JSR [CHECK.RESULT] ;CHECK IDENTIFIER CODE
U 17FB, 097F,14 :8024      JMP [LOOP.T8.9] ;LOOP ON ERROR IF ENABLED
U 17FC, 8870,0C :8025      JSR [INC.EN] ;INCREMENT ERROR NUMBER TO A
:8026
U 17FD, 3660,15 :8027      MOV LS[INTERUPT.EN] TO WR[0]
U 17FE, C762,15 :8028      BIS LS[CONSOLE.HALT] TO WR[0]
U 17FF, 3E80,15 :8029      MOV WR[0] TO LS[CONTROL.STATUS] ;CSR = BIT16:BIT17
U 1800, 10E0,15 :8030      MISC [SET.CP.ATTN] ;CALL CONSOLE
:8031
U 1801, 0980,14 :8032      WAIT.T8.A: JMP [WAIT.T8.A] ;ASSERT CONS HALT
:8033
U 1802, B648,95 :8034      LOOP.T8.A: MOV LS[BIT4] TO WR[1]
U 1803, 4746,95 :8035      BIS LS[BIT3] TO WR[1]
U 1804, C744,95 :8036      BIS LS[BIT2] TO WR[1] ;EXPECTED DATA = 0111(B) IN BITS 2-5
U 1805, 36FC,15 :8037      MOV LS[INTERUPT.VEC] TO WR[0] ;GET IDENTIFIER CODE
U 1806, 0869,3C :8038      JSR [CHECK.RESULT] ;CHECK IDENTIFIER CODE
U 1807, 0980,24 :8039      JMP [LOOP.T8.A] ;LOOP ON ERROR IF ENABLED
U 1808, 0870,8C :8040      JSR [ASSERT.NA]
U 1809, B724,15 :8041      MOV LS[HI.IPL] TO WR[0]
U 180A, C77C,15 :8042      BIS LS[BIT30] TO WR[0] ;WRO = 40 1F 00 00
U 180B, BFFE,15 :8043      MOV WR[0] TO LS[PSL.HW] ;LOAD IPL AND ENABLE T-TRAP
:8044
U 180C, 90FC,15 :8045      LOOP.T8.B: MISC2 [MASK.HALT.AND.T.BIT] ;MASK T-TRAP & HALT INTERRUPTS
U 180D, DB00,07 :8046      SKIP.IF[NO.INTERUPT] ;SHOULD SKIP
U 180E, 8981,74 :8047      JMP [T8.ERROR.B] ;GO TO ERROR CALL
:8048
U 180F, DB00,07 :8049      T8.C: SKIP.IF[NO.INTERUPT] ;SHOULD NOT SKIP
U 1810, 5800,1E :8050      SKIP
U 1811, 0981,F4 :8051      JMP [T8.ERROR.C] ;GO TO ERROR CALL
:8052
U 1812, DB0C,15 :8053      LOOP.T8.D: MOV LS[6] TO Q ;THIS INSTRUCTION TESTS THE MISC DECODER
:8054 ;BITS 18, AND 11-9 ARE SAME AS MASK T-BIT
U 1813, DB00,07 :8055      SKIP.IF[NO.INTERUPT] ;SHOULD NOT SKIP
U 1814, 5800,1E :8056      SKIP
U 1815, 8982,74 :8057      JMP [T8.ERROR.D] ;GO TO ERROR CALL
U 1816, 0982,F4 :8058      JMP [END.T8]
:8059
:8060
U 1817, B646,15 :8061      T8.ERROR.B: MOV LS[#8] TO WR[0]
U 1818, C0C0,15 :8062      ADD LS[#3(W)] TO WR[0]
  
```

```

U 1819, BE82,15 :8063      MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR NUMBER = B
U 181A, 2F80,15 :8064      CLR WR[0]
U 181B, 369E,95 :8065      MOV LS[ONES] TO WR[1] ;SET BAD DATA
U 181C, 0869,3C :8066      JSR [CHECK.RESULT] ;REPORT ERROR
U 181D, 8980,C4 :8067      JMP [LOOP.T8.B] ;LOOP ON ERROR IF ENABLED
U 181E, 8980,F4 :8068      JMP [T8.C] ;GO TO NEXT PART OF TEST
:8069
:8070
U 181F, B646,15 :8071      T8.ERROR.C:
U 1820, C044,15 :8072      MOV LS[#8] TO WR[0]
U 1821, BE82,15 :8073      ADD LS[#4] TO WR[0]
U 1822, 2F80,15 :8074      MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR NUMBER = C
U 1823, 369E,95 :8075      CLR WR[0]
U 1824, 0869,3C :8076      MOV LS[ONES] TO WR[1] ;SET BAD DATA
U 1825, 8980,C4 :8077      JSR [CHECK.RESULT] ;REPORT ERROR
U 1826, 8981,24 :8078      JMP [LOOP.T8.B] ;LOOP ON ERROR IF ENABLED
:8079      JMP [LOOP.T8.D] ;GO DO PART D OF TEST
:8080
U 1827, B646,15 :8081      T8.ERROR.D:
U 1828, C044,15 :8082      MOV LS[#8] TO WR[0]
U 1829, 4040,15 :8083      ADD LS[#4] TO WR[0]
U 182A, BE82,15 :8084      ADD LS[#1] TO WR[0]
U 182B, 2F80,15 :8085      MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR NUMBER = D
U 182C, 369E,95 :8086      CLR WR[0]
U 182D, 0869,3C :8087      MOV LS[ONES] TO WR[1] ;SET BAD DATA
U 182E, 8981,24 :8088      JSR [CHECK.RESULT] ;REPORT ERROR
:8089      JMP [LOOP.T8.D] ;LOOP ON ERROR IF ENABLED
:8090
:8091
U 182F, B724,15 :8092      END.T8:
U 1830, BFFE,15 :8093      MOV LS[HI.IPL] TO WR[0]
U 1831, 3660,15 :8094      MOV WR[0] TO LS[PSL.HW] ;SET IPI TO 1F
U 1832, 3E80,15 :8095      MOV LS[INTERUPT.EN] TO WR[0] ;SET B OF WR
U 1833, 10E0,15 :8096      MOV WR[0] TO LS[CONTROL.STATUS] ;SET U J CLEAR ALL CONSOLE INTERRUPTS
:8097      MISC [SET.CP.ATTN] ;CALL CONSOLE
U 1834, 0983,44 :8098      WAIT.T8.END:
:8099      JMP [WAIT.T8.END] ;DEASSERT CONSOLE INTERRUPTS

```

:8100
:8101
:8102
:8103
:8104
:8105
:8106
:8107
:8108
:8109
:8110
:8111
:8112
:8113
:8114
:8115
:8116
:8117
:8118
:8119
:8120
:8121
:8122
:8123
:8124
:8125
:8126
:8127
:8128
:8129
:8130
:8131
:8132
:8133
:8134
:8135
:8136
:8137
:8138
:8139
:8140
:8141
:8142
:8143
:8144
:8145
:8146
:8147
:8148
:8149
:8150
:8151
:8152
:8153
:8154

Page "TEST 9 - SHIFT DATA PAL SI=ASH.WR Test (M8390 CPU Module)"

++

Test Description:

This test checks the SHIFT DATA PAL for SI=ASH.WR (SI=2). The macro instructions ASHL WR and ASHR WR are used for this test. These instructions use the same 2901 control signals as ROL and ROR and have been previously tested. The SHIFT DATA PAL is responsible for inserting a 0 into bit 0 on a ASHL instruction, and performing a sign extend on a ASHR instruction. The data path is as follows: WR 0 is loaded with data from LS. Then a ASHL WR[0] or ASHR WR[0] is executed, WR 1 is loaded with expected data, and the result is checked. Data patterns used are various patterns of 1's and 0's in bits 31 and 0 of the WR.

Assumptions:

It is assumed that the previous 2901 shift tests have run successfully.

Test Steps:

- 1) Set up error mask, error number, and module number in LS (for error printouts) and clear previous error number in LS.
- 2) Load WR 0 with 1's in bits 31 and 0 and execute an ASHL WR[0]. Check WR 0 for a 2(H).
- 3) Load WR 0 with a 1 in bit 31 and execute an ASHR WR[0]. Check WR 0 for a 00 00 00 00(H) (sign extend a negative number).
- 4) Load WR 0 with a 1 in bit 0 and execute ASHR WR[0] and check for a zero (sign extend a positive number).

Errors:

- Error 1 - ASHL failed with data of 1 in bits 31 and 0.
Error 2 - ASHR failed with data of 1 in bit 31.
Error 3 - ASHR failed with data of 1 in bit 0.

Debug:

Error 1 - This error most likely indicates a problem with the SHIFT DATA PAL itself. During the ASHL instruction, check the inputs for a 2(H) on CSR 13-11. DEST CTL 1 H and DEST CTL 2 H should both be high. This is all that is necessary to get a low on the output RAM SHF LSB H. If the inputs are OK but the output is high, then the PAL is bad.

Error 2 - This error most likely indicates a problem with the SHIFT DATA PAL or the N LONG H input. During the ASHR instruction, check the inputs for a 2(H) on CSR 13-11, a low on DEST CTL 1 H, a high on DEST CTL 2 H, and a high on N LONG H. The output RAM SHF MSB H should be high. If the inputs are OK, suspect the SHIFT DATA PAL.

Error 3 - This error most likely indicates a problem with the SHIFT DATA PAL or the N LONG H input. During the ASHR instruction, check

```

:8155      : the inputs for a 2(H) on CSR 13-11, a low on DEST CTL 1 H, a high on
:8156      : DEST CTL 2 H, and a low on N LONG H. The output RAM SHF MSB H should
:8157      : be low. If the inputs are OK, suspect the SHIFT DATA PAL.
:8158      :
:8159      :
:8160      :
U 1835, B65E,15 :8161      T.9:--
U 1836, 3E80,15 :8162      MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT15 IN WRO FOR CONTROL AND STATUS WORD
:8163      MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT15 IN CNTRL AND STATUS WORD
:8164      MISC [SET.CP.ATTN] ;BIT15 INDICATES START OF TEST TO CONSOLE
:8165      WAIT.T9.0: ;ISSUE CPU ATTN TO CONSOLE
U 1838, 0983,84 :8166      JMP [WAIT.T9.0] ;LOOP HERE WAITING FOR CONSOLE TO RESPOND
U 1839, E58A,15 :8167      CLR LS[ERROR.MASK] ;CLEAR ERROR MASK.
U 183A, 65A0,15 :8168      CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
U 183B, B640,15 :8169      MOV LS[#1] TO WR[0] ;SET WR 0 TO 1
U 183C, BE82,15 :8170      MOV WR[0] TO LS[ERROR.NUMBER] ;SET ERROR NUMBER TO FIRST ERROR
U 183D, A3C0,15 :8171      ROL WR[0] ;SET WRO TO 2
U 183E, 3E8C,15 :8172      MOV WR[0] TO LS[MODULE.NUM] ;SET UP MODULE CODE FOR CPU MODULE
:8173      LOOP.T9.1:
U 183F, 367E,15 :8174      MOV LS[BIT31] TO WR[0]
U 1840, C740,15 :8175      BIS LS[BIT0] TO WR[0] ;WRO = 80 00 00 01
U 1841, 23D0,15 :8176      ASHL WR[0] ;SHIFT LEFT
U 1842, B642,95 :8177      MOV LS[BIT1] TO WR[1] ;EXPECTED = 00 00 00 02
U 1843, 0869,3C :8178      JSR [CHECK.RESULT]
U 1844, 8983,F4 :8179      JMP [LOOP.T9.1] ;LOOP ON ERROR IF ENABLED
U 1845, 8870,0C :8180      JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 2
:8181      LOOP.T9.2:
U 1846, 367E,15 :8182      MOV LS[BIT31] TO WR[0] ;WRO = 80 00 00 00
U 1847, A350,15 :8183      ASHR WR[0] ;SHIFT RIGHT
U 1848, B67E,95 :8184      MOV LS[BIT31] TO WR[1]
U 1849, 477C,95 :8185      BIS LS[BIT30] TO WR[1] ;EXPECTED = C0 00 00 00
U 184A, 0869,3C :8186      JSR [CHECK.RESULT]
U 184B, 0984,64 :8187      JMP [LOOP.T9.2] ;LOOP ON ERROR IF ENABLED
U 184C, 8870,0C :8188      JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 3
:8189      LOOP.T9.3:
U 184D, B640,15 :8190      MOV LS[BIT0] TO WR[0] ;WRO = 00 00 00 01
U 184E, A350,15 :8191      ASHR WR[0] ;SHIFT RIGHT
U 184F, 2F82,95 :8192      CLR WR[1] ;EXPECTED = 00 00 00 00
U 1850, 0869,3C :8193      JSR [CHECK.RESULT]
U 1851, 8984,D4 :8194      JMP [LOOP.T9.3] ;LOOP ON ERROR IF ENABLED
:8195      END.T9:
  
```

:8196 Page "TEST A - SHIFT DATA PAL SI=ASH.WR.Q Test (M8390 CPU Module)"
:8197 ++

:8198
:8199 Test Description:

:8200
:8201 This test checks the SHIFT DATA PAL for SI=ASH.WR.Q (SI=3). The macro
:8202 instructions ASHLC WR.Q and ASHRC WR.Q are used for this test. These
:8203 instructions use the same 2901 control signals as ROLC and RORC and
:8204 have been previously tested. The SHIFT DATA PAL is responsible for
:8205 inserting a 0 into the Q reg LSB (the Q reg is the lower 32 bits of
:8206 the 64 bit quadword) and loading the RAM LSB from the Q MSB on a ASHLC
:8207 instruction. On a ASHRC, the SHIFT DATA PAL is responsible for sign
:8208 extending into the RAM MSB (duplicating the RAM MSB contents before
:8209 the shift) and loading the Q MSB from the RAM LSB. Data patterns used
:8210 will be various patterns of 1's and 0's in bits 31 and 0 of the WR and
:8211 the Q reg.
:8212

:8213 Assumptions:

:8214 It is assumed that the previous 2901 shift tests have run successfully.
:8215

:8216 Test Steps:

- :8217
:8218 1) Set up error mask, error number, and module number in LS (for
:8219 error printouts) and clear previous error number in LS.
:8220 2) Load WR 0 and the Q reg with 1's in bits 31 and 0, 0's in other
:8221 bits.
:8222 3) Execute ASHLC WR[0].Q and check WR 0 for a 3(H).
:8223 4) Load WR 0 with Q reg and check for a 2(H). This tests 0 fill in
:8224 bottom bit.
:8225 5) Load WR 0 with 0's and the Q reg with a 1 in bit 31.
:8226 6) Execute ASHLC WR[0].Q and check WR 0 for a 1(H).
:8227 7) Load WR 0 with Q reg and check for a 0.
:8228 8) Load WR 0 with a 1 in bit 31, other bits 0 and load the Q reg with
:8229 all 0's.
:8230 9) Execute ASHRC WR[0].Q and check WR 0 for a C0 00 00 00(H). This
:8231 tests sign extend with a negative number.
:8232 10) Load WR 0 with Q reg and check for a 0.
:8233 11) Load WR 0 with all 0's and load the Q reg with a 1 in bit 0.
:8234 12) Execute ASHRC WR[0].Q and check WR 0 for a 0. This tests sign ex-
:8235 tend with a positive number.
:8236 13) Load WR 0 with Q reg and check for a 0.
:8237 14) Load WR 0 with a 1 in bit 0, other bits 0 and load Q reg with all
:8238 0's.
:8239 15) Execute ASHRC WR[0].Q and check WR 0 for a 0.
:8240 16) Load WR 0 with Q reg and check for a 80 00 00 00. This checks the
:8241 Q reg MSB load.
:8242
:8243

:8244 Errors:

- :8245
:8246 Error 1 - ASHLC WR[0].Q failed to shift WR 0 correctly. Data pattern
:8247 before shift was 1's in bits 31 and 0 of WR 0 and the Q reg.
:8248 Error 2 - ASHLC WR[0].Q failed to shift the Q reg correctly. Data
:8249 before shift was 1's in bits 31 and 0 of WR 0 and the Q reg.
:8250 Error 3 - ASHLC WR[0].Q failed to shift WR 0 correctly. Data pattern

22-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

G 15
TEST A - SHIFT DATA 3-MAR-82 FICHE 1 Frame G15
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Sequence 188
TEST A - SHIFT DATA PAL SI=ASH.WR.Q Test (M8390 CPU Module) Rev 01.00 Page 182

:8251 : before shift was a 1 in bit 31 of the Q reg, 0's in all
:8252 : other bits.
:8253 : Error 4 - ASHLC WR[0].Q failed to shift the Q reg correctly. Data
:8254 : before shift was a 1 in bit 31 of the Q reg, 0's in all
:8255 : other bits.
:8256 : Error 5 - ASHRC WR[0].Q failed to shift WR 0 correctly. Data pattern
:8257 : before shift was a 1 in bit 31 of WR 0, 0's in all other
:8258 : bits.
:8259 : Error 6 - ASHRC WR[0].Q failed to shift the Q reg correctly. Data
:8260 : before shift was a 1 in bit 31 of WR 0, 0's in all other
:8261 : bits.
:8262 : Error 7 - ASHRC WR[0].Q failed to shift WR 0 correctly. Data pattern
:8263 : before shift was a 1 in bit 0 of the Q reg, 0's in all other
:8264 : bits.
:8265 : Error 8 - ASHRC WR[0].Q failed to shift the Q reg correctly. Data
:8266 : before shift was a 1 in bit 0 of the Q reg, 0's in all other
:8267 : bits.
:8268 : Error 9 - ASHRC WR[0].Q failed to shift WR 0 correctly. Data pattern
:8269 : before shift was a 1 in bit 0 of WR 0, 0's in all other
:8270 : bits.
:8271 : Error A- ASHRC WR[0].Q failed to shift the Q reg correctly. Data
:8272 : before shift was a 1 in bit 0 of WR 0, 0's in all other
:8273 : bits.
:8274 :

:8275 : Debug:

:8276 :
:8277 : Error 1 - This error indicates a problem with the SHIFT DATA PAL.
:8278 : The output RAM SHF LSB H should be high during the shift u-word.
:8279 : The inputs to the SHIFT DATA PAL should be a high on Q SHF MSB H
:8280 : and a 3(H) on CSR 13-11. If these inputs are OK but the output
:8281 : is low, the PAL is bad.
:8282 :
:8283 : Error 2 - This error indicates a problem with the SHIFT DATA PAL.
:8284 : The output Q SHF LSB H should be low during the shift u-word. If
:8285 : not, check CSR 13-11 for a 3(H). This is all that is necessary
:8286 : to make the output low.
:8287 :
:8288 : Error 3 - Same as error 1.
:8289 :
:8290 : Error 4 - Same as error 2.
:8291 :
:8292 : Error 5 - This error indicates a problem with the SHIFT DATA PAL.
:8293 : The output RAM SHF MSB H should be high during the shift u-word. If
:8294 : not, check the inputs for a 3(H) on CSR 13-11 and a high on N LONG H.
:8295 : The PAL itself is bad if the inputs are OK but the output is low.
:8296 :
:8297 : Error 6 - This error indicates a problem with the SHIFT DATA PAL.
:8298 : The output Q SHF MSB H should be low during the shift u-word. If not,
:8299 : Check the inputs for a 3(H) on CSR 13-11 and a low on RAM SHF LSB H.
:8300 : The PAL is bad if the inputs are OK but the output is high.
:8301 :
:8302 : Error 7 - This error indicates a problem with the SHIFT DATA PAL.
:8303 : The output RAM SHF MSB H should be low during the shift u-word. If
:8304 : not, check the inputs for a 3(H) on CSR 13-11 and a low on N LONG H.
:8305 : The PAL itself is bad if the inputs are OK but the output is low.

```

:8306
:8307
:8308
:8309
:8310
:8311
:8312
:8313
:8314
:8315
:8316
:8317
:8318
:8319
:8320
:8321
:8322
:8323
:8324
:8325
:8326
:8327
:8328
:8329
:8330
:8331
:8332
:8333
:8334
:8335
:8336
:8337
:8338
:8339
:8340
:8341
:8342
:8343
:8344
:8345
:8346
:8347
:8348
:8349
:8350
:8351
:8352
:8353
:8354
:8355
:8356
:8357
:8358
:8359
:8360
  
```

Error 8 - same as error 6.
 Error 9 - Same as error 7.
 Error A - This error indicates a problem with the SHIFT DATA PAL.
 The output Q SHF MSB H should be high during the shift u-word. If not,
 Check the inputs for a 3(H) on CSR 13-11 and a high on RAM SHF LSB H.
 The PAL is bad if the inputs are OK but the output is high.

--

T.A:

```

U 1852, B65E,15 :8318 MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT15 IN WRO FOR CONTROL AND STATUS WORD
U 1853, 3E80,15 :8319 MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT15 IN CNTRL AND STATUS WORD
:8320 ; BIT15 INDICATES START OF TEST TO CONSOLE
U 1854, 10E0,15 :8321 MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:8322
WAIT.TA.0:
U 1855, 8985,54 :8323 JMP [WAIT.TA.0] ;LOOP HERE WAITING FOR CONSOLE TO RESPOND
U 1856, E58A,15 :8324 CLR LS[ERROR.MASK] ;CLEAR ERROR MASK.
U 1857, 65A0,15 :8325 CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
U 1858, 3642,15 :8326 MOV LS[CPU] TO WR[0] ;SET BIT 1 IN WR
U 1859, 3E8C,15 :8327 MOV WR[0] TO LS[MODULE.NUM] ;SET UP MODULE CODE FOR CPU MODULE
:8328
LOOP.TA.1:
U 185A, B640,15 :8329 MOV LS[#1] TO WR[0] ;SET WR 0 TO 1
U 185B, BE82,15 :8330 MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR NUMBER = 1
U 185C, 367E,15 :8331 MOV LS[BIT31] TO WR[0]
U 185D, C740,15 :8332 BIS LS[BIT0] TO WR[0] ;WRO = 80 00 00 01
U 185E, D87E,15 :8333 MOV LS[BIT31] TO Q
U 185F, D340,15 :8334 BIS LS[BIT0] TO Q ; Q = 80 00 00 01
U 1860, 2398,15 :8335 ASHLQ WR[0],Q ;SHIFT LEFT THE QUAD WORD [WRO-Q]
U 1861, 7610,15 :8336 MOV Q TO LS[T8] ;STORE Q
U 1862, B6C0,95 :8337 MOV LS[#3(H)] TO WR[1] ;EXPECTED IN WRO = 00 00 00 03
U 1863, 0869,3C :8338 JSR [CHECK.RESULT] ;CHECK
U 1864, 8985,A4 :8339 JMP [LOOP.TA.1] ;LOOP ON ERROR IF ENABLED
U 1865, 8870,0C :8340 JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 2
:8341
TA.2:
U 1866, B610,15 :8342 MOV LS[T8] TO WR[0] ;GET Q
U 1867, B642,95 :8343 MOV LS[#2] TO WR[1] ;EXPECTED IN Q = 00 00 00 02
U 1868, 0869,3C :8344 JSR [CHECK.RESULT] ;CHECK
U 1869, 8985,A4 :8345 JMP [LOOP.TA.1] ;LOOP ON ERROR IF ENABLED
:8346
LOOP.TA.3:
U 186A, 36C0,15 :8347 MOV LS[#3(H)] TO WR[0]
U 186B, BE82,15 :8348 MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR NUMBER = 3
U 186C, D87E,15 :8349 MOV LS[BIT31] TO Q ; Q = 80 00 00 00
U 186D, 2F80,15 :8350 CLR WR[0] ;WRO = 00 00 00 00
U 186E, 2398,15 :8351 ASHLQ WR[0],Q ;SHIFT LEFT
U 186F, 7610,15 :8352 MOV Q TO LS[T8] ;STORE Q
U 1870, 3640,95 :8353 MOV LS[#1] TO WR[1] ;EXPECTED (WRO) = 00 00 00 01
U 1871, 0869,3C :8354 JSR [CHECK.RESULT]
U 1872, 8986,A4 :8355 JMP [LOOP.TA.3] ;LOOP ON ERROR IF ENABLED
U 1873, 8870,0C :8356 JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 4
:8357
TA.4:
U 1874, B610,15 :8358 MOV LS[T8] TO WR[0] ;GET Q
U 1875, 2F82,95 :8359 CLR WR[1] ;EXPECTED (Q) = 00 00 00 00
U 1876, 0869,3C :8360 JSR [CHECK.RESULT]
  
```

I 15

ZZ-ENKCB-1.0 : ENKCB.MIC TEST A - SHIFT DATA 3-MAR-82 Fiche 1 Frame 115 Sequence 190
 : ENKCB.MIC MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Rev 01.00 Page 184
 : ENKCB.MIC TEST A - SHIFT DATA PAL SI=ASH.WR.Q Test (M8590 CPU Module)

```

U 1877. 8986.A4 :8361      JMP [LOOP.TA.3]                ;LOOP ON ERROR IF ENABLED
:8362
U 1878. 3644.15 :8363      LOOP.TA.5:
U 1879. 2040.15 :8364      MOV LS[#4] TO WR[0]
U 187A. BE82.15 :8365      INC WR[0]
U 187B. 367E.15 :8366      MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR NUMBER = 5
U 187C. D89C.15 :8367      MOV LS[BIT31] TO WR[0] ;WRO = 80 00 00 00
U 187D. A318.15 :8368      MOV LS[ZERO] TO Q ;Q = 00 00 00 00
U 187E. 7610.15 :8369      ASHRC WR[0],Q ;SHIFT RIGHT
U 187F. B67E.95 :8370      MOV Q TO LS[T8] ;STORE Q
U 1880. 477C.95 :8371      MOV LS[BIT31] TO WR[1]
U 1881. 0869.3C :8372      BIS LS[BIT30] TO WR[1] ;EXPECTED (WRO) = C0 00 00 00
U 1882. 8987.84 :8373      JSR [CHECK.RESULT]
U 1883. 8870.0C :8374      JMP [LOOP.TA.5] ;LOOP ON ERROR IF ENABLED
:8375      JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 6
U 1884. B610.15 :8376      TA.6:
U 1885. 2F82.95 :8377      MOV LS[T8] TO WR[0] ;GET Q
U 1886. 0869.3C :8378      CLR WR[1] ;EXPECTED (Q) = 00 00 00 00
U 1887. 8987.84 :8379      JSR [CHECK.RESULT]
:8380      JMP [LOOP.TA.5] ;LOOP ON ERROR IF ENABLED
U 1888. B646.15 :8381      LOOP.TA.7:
U 1889. 2100.15 :8382      MOV LS[#8] TO WR[0]
U 188A. BE82.15 :8383      DEC WR[0]
U 188B. 2F80.15 :8384      MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR NUMBER = 7
U 188C. 5840.15 :8385      CLR WR[0] ;WRO = 00 00 00 00
U 188D. A318.15 :8386      MOV LS[#1] TO Q ;Q = 00 00 00 01
U 188E. 7610.15 :8387      ASHRC WR[0],Q ;SHIFT RIGHT
U 188F. 2F82.95 :8388      MOV Q TO LS[T8] ;STORE Q
U 1890. 0869.3C :8389      CLR WR[1] ;EXPECTED (WRO) = 00 00 00 00
U 1891. 8988.84 :8390      JSR [CHECK.RESULT]
U 1892. 8870.0C :8391      JMP [LOOP.TA.7] ;LOOP ON ERROR IF ENABLED
:8392      JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 8
U 1893. B610.15 :8393      TA.8:
U 1894. 2F82.95 :8394      MOV LS[T8] TO WR[0] ;GET STORED VALUE OF Q
U 1895. 0869.3C :8395      CLR WR[1] ;EXPECTED (Q) = 00 00 00 00
U 1896. 8988.84 :8396      JSR [CHECK.RESULT]
:8397      JMP [LOOP.TA.7] ;LOOP ON ERROR IF ENABLED
U 1897. B646.15 :8398      LOOP.TA.9:
U 1898. 2040.15 :8399      MOV LS[#8] TO WR[0]
U 1899. BE82.15 :8400      INC WR[0]
U 189A. B640.15 :8401      MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR NUMBER = 9
U 189B. D89C.15 :8402      MOV LS[#1] TO WR[0] ;WRO = 00 00 00 01
U 189C. A318.15 :8403      MOV LS[ZERO] TO Q ;Q = 00 00 00 00
U 189D. 7610.15 :8404      ASHRC WR[0],Q ;SHIFT RIGHT
U 189E. 2F82.95 :8405      MOV Q TO LS[T8] ;STORE Q
U 189F. 0869.3C :8406      CLR WR[1] ;EXPECTED (WRO) = 00 00 00 00
U 18A0. 0989.74 :8407      JSR [CHECK.RESULT]
U 18A1. 8870.0C :8408      JMP [LOOP.TA.9] ;LOOP ON ERROR IF ENABLED
:8409      JSR [INC.EN] ;INCREMENT ERROR NUMBER TO A
U 18A2. B610.15 :8410      TA.A:
U 18A3. B67E.95 :8411      MOV LS[T8] TO WR[0] ;GET STORED VALUE OF Q
U 18A4. 0869.3C :8412      MOV LS[BIT31] TO WR[1] ;EXPECTED (Q) = 80 00 00 00
U 18A5. 0989.74 :8413      JSR [CHECK.RESULT]
:8414      JMP [LOOP.TA.9] ;LOOP ON ERROR IF ENABLED
END.TA:
  
```


ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

J 15
TEST B - SHIFT DATA 3-MAR-82 FICHE 1 Frame J15
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Rev 01.00 Sequence 191
TEST B - SHIFT DATA PAL SI=SHF.WR.Q Test (M8390 CPU Module) Page 185

8415 Page "TEST B - SHIFT DATA PAL SI=SHF.WR.Q Test (M8390 CPU Module)"
8416 ++
8417
8418 Test Description:
8419
8420 This test checks the SHIFT DATA PAL for SI=SHF.WR.Q (SI=6). The macro
8421 instruction SHRC WR.Q is used for this test. These instruction uses
8422 the same 2901 control signals as ASHRC and has been previously tested.
8423 The SHIFT DATA PAL is responsible for inserting a 0 into bit 31 of the
8424 WR. The data path is as follows: WR 0 is loaded with data from LS.
8425 Then a SHRC WR[0].Q is executed, WR 1 is loaded with expected data,
8426 and the result is checked. Data patterns used are various patterns
8427 of 1's and 0's in bits 31 and 0 of the WR.
8428
8429
8430 Assumptions:
8431
8432 It is assumed that the previous 2901 shift tests have run successfully.
8433
8434 Test Steps:
8435
8436 1) Set up error mask, error number, and module number in LS (for
8437 error printouts) and clear previous error number in LS.
8438 2) Load WR 0 with a 1 in bit 31 and the Q reg with 1's in bits 31 and
8439 0 and execute a SHRC WR[0].Q. Check WR 0 for a 40 00 00 00(H) and
8440 the Q reg for 40 00 00 00(H).
8441 3) Load WR 0 with a 1 in bit 0 and the Q reg with 0's. Execute SHRC
8442 WR[0].Q and check for a 0 in WR 0 and a 80 00 00 00(H) in the Q
8443 reg.
8444
8445 Errors:
8446
8447 Error 1 - WR 0 failed on SHRC WR.Q with data of 1 in bit 31 of WR 0,
8448 and a 1 in bits 31 and 0 of the Q reg.
8449 Error 2 - Q reg failed with above data.
8450 Error 3 - WR 0 failed on SHRC WR.Q with data of 1 in bit 0 WR 0 and
8451 0's in the Q reg.
8452 Error 4 - Q reg failed with above data.
8453
8454 Debug:
8455
8456 Error 1 - This error indicates a problem with the SHIFT DATA PAL.
8457 Check the inputs to this PAL during the SHRC instruction for a 6(H)
8458 on CSR bits 13-11. The output RAM SHF MSB H should be low. If not,
8459 the PAL is bad.
8460
8461 Error 2 - Same as error 1, except the output Q SHF MSB H should be
8462 low.
8463
8464 Error 3 - Same as error 1.
8465
8466 Error 4 - Same as error 1, except the output Q SHF MSB H should be
8467 high.
8468
8469 --

ZZ-ENKCB-1.0 : ENKCB.MIC
: ENKCB.MCR
: ENKCB.MIC

K 15
TEST B - SHIFT DATA 3-MAR-E2
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Rev 01.00
TEST B - SHIFT DATA PAL SI=SHF.WR.Q Test (M8390 CPU Module)

Fiche 1 Frame K15

Sequence 192

Page 186

```
:8470
U 18A6, B65E,15 :8471
U 18A7, 3E80,15 :8472
:8473
U 18A8, 10E0,15 :8474
:8475
L 18A9, 898A,94 :8476
U 18AA, E58A,15 :8477
U 18AB, 65A0,15 :8478
U 18AC, 3642,15 :8479
U 18AD, 3E8C,15 :8480
:8481
U 18AE, B640,15 :8482
U 18AF, BE82,15 :8483
U 18B0, 367E,15 :8484
U 18B1, D87E,15 :8485
U 18B2, D340,15 :8486
U 18B3, A330,15 :8487
U 18B4, 7610,15 :8488
U 18B5, 367C,95 :8489
U 18B6, 0869,3C :8490
U 18B7, 098A,E4 :8491
U 18B8, 8870,0C :8492
:8493
U 18B9, B610,15 :8494
U 18BA, 367C,95 :8495
U 18BB, 0869,3C :8496
U 18BC, 098A,E4 :8497
:8498
U 18BD, 36C0,15 :8499
U 18BE, BE82,15 :8500
U 18BF, B640,15 :8501
U 18C0, D89C,15 :8502
U 18C1, A330,15 :8503
U 18C2, 7610,15 :8504
U 18C3, 2F82,95 :8505
U 18C4, 0869,3C :8506
U 18C5, 898B,D4 :8507
U 18C6, 8870,0C :8508
:8509
U 18C7, B610,15 :8510
U 18C8, B67E,95 :8511
U 18C9, 0869,3C :8512
U 18CA, 898B,D4 :8513
:8514

T.B:
    MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT15 IN WR0 FOR CONTROL AND STATUS WORD
    MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT15 IN CNTROL AND STATUS WORD
    MISC [SET.CP.ATTN] ;BIT15 INDICATES START OF TEST TO CONSOLE
    ;ISSUE CPU ATTN TO CONSOLE
WAIT.TB.0:
    JMP [WAIT.TB.0] ;LOOP HERE WAITING FOR CONSOLE TO RESPOND
    CLR LS[ERROR.MASK] ;CLEAR ERROR MASK.
    CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
    MOV LS[CPU] TO WR[0] ;SET BIT 1 OF WR 0
    MOV WR[0] TO LS[MODULE.NUM] ;SET UP MODULE CODE FOR CPU MODULE
LOOP.TB.1:
    MOV LS[#1] TO WR[0] ;SET WR 0 TO 1
    MOV WR[0] TO LS[ERROR.NUMBER] ;SET ERROR NUMBER = 1
    MOV LS[BIT31] TO WR[0] ;WRO = 80 00 00 00
    MOV LS[BIT31] TO Q ;
    BIS LS[BIT0] TO Q ; Q = 80 00 00 01
    SHRC WR[0],Q
    MOV Q TO LS[T8] ;STORE Q
    MOV LS[BIT30] TO WR[1] ;EXPECTED (WRO) = 40 00 00 00
    JSR [CHECK.RESULT]
    JMP [LOOP.TB.1] ;LOOP ON ERROR IF ENABLED
    JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 2
TB.2:
    MOV LS[T8] TO WR[0] ;GET STORED VALUE OF Q
    MOV LS[BIT30] TO WR[1] ;EXPECTED (Q) = 40 00 00 00
    JSR [CHECK.RESULT]
    JMP [LOOP.TB.1] ;LOOP ON ERROR IF ENABLED
LOOP.TB.3:
    MOV LS[#3(H)] TO WR[0]
    MOV WR[0] TO LS[ERROR.NUMBER] ;SET ERROR NUMBER = 3
    MOV LS[#1] TO WR[0] ;WRO = 00 00 00 01
    MOV LS[ZERO] TO Q ; Q = 00 00 00 00
    SHRC WR[0],Q
    MOV Q TO LS[T8] ;STORE Q
    CLR WR[1] ;EXPECTED (WRO) = 00 00 00 00
    JSR [CHECK.RESULT]
    JMP [LOOP.TB.3] ;LOOP ON ERROR IF ENABLED
    JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 4
TB.4:
    MOV LS[T8] TO WR[0] ;GET STORED VALUE OF Q
    MOV LS[BIT31] TO WR[1] ;EXPECTED (Q) = 80 00 00 00
    JSR [CHECK.RESULT]
    JMP [LOOP.TB.3] ;LOOP ON ERROR IF ENABLED
END.TB:
```

Page "TEST C - Signed Multiply (SI=MUL) Test (M8390 CPU Module)"

Test Description:

This test checks the conditional add and shift logic. The instruction COND.ADD&SHIFT WR TO WR.Q is used in this test. This is the instruction used for a signed or unsigned hardware multiply. What makes this instruction different from the normal 2901 control is that the SRC mode is dependent on the value of the Q reg bit 0 before a previous shift operation. The control signals for the 2901 during the conditional instruction are as follows: SRC=MUL FUNC=R PLUS S DST=RSHF.RAM.Q CIN=NO CIN. the SRC mode will be either A.B, if the Q reg contained a 1 in bit 0 before a previous shift, or 0.B if the Q reg had a 0 in bit 0 before a previous shift. The SI field is set to a 4 for this test. This test checks the operation of COND.ADD&SHIFT WR TO WR.Q with both previous shift conditions. The test first loads 2 WR's and the Q reg with data and shifts the Q reg right to load MPLIER LSB H, which controls the SRC mode for the next instruction. Then a COND.ADD&SHIFT WR[2] TO WR[0].Q instruction is executed and the result is checked. Data patterns are picked that will check the shift and the XOR of N and V that goes into the WR MSB. Another instruction, COND.SUB&SHIFT WR FROM WR.Q, is executed also to check the DEST CTL ROM and SRC.ALU CTL PAL. The control signals for this instruction are as follows: SRC= MUL FUNC=S MINUS R DST=RSHF.RAM.Q CIN=CIN. Only one pattern will be used to check this function as the only difference from the conditional ADD is in the function of the 2901.

Assumptions:

It is assumed that the previous shift data PAL tests have run successfully.

Test Steps:

- 1) Set up error mask, error number, and module number in LS (for error printouts) and clear previous error number in LS.
- 2) Load WR 2, WR 0, and the Q reg with a 1. Load WR 3 with all 0's, and execute ASHRC WR[3].Q. This loads MPLIER LSB H with a 1 and leaves 0 in the Q reg.
- 3) Execute COND.ADD&SHIFT WR[2] to WR[0].Q. The MPLIER LSB H signal being high will cause an A PLUS B (1 plus 1). The shift will put a 1 in bit 0 of WR 0, and the Q reg will still be 0. Bit 31 of WR 0 will be 0 because N LONG and V LONG are both clear. Check the results for these operands.
- 4) Load WR 2 and WR 0 with a 1, clear the Q reg, and execute ASHR WR[3].Q to load MPLIER LSB H with a 0.
- 5) Execute COND.ADD&SHIFT WR[2] to WR[0].Q. The MPLIER LSB H signal being low will cause an 0 PLUS B (0 plus 1). The shift will put a 1 in bit 31 of the Q reg, and WR 0 will be 0. Bit 31 of WR 0 will be 0 because N LONG and V LONG are both clear. Check the results for these operands.
- 6) Repeat step 2 except load WR 2 with 80 00 00 01 (negative number).

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

M 15
TEST C - Signed Mul 3-MAR-82 FICHE 1 Frame M15
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Sequence 194
TEST C - Signed Multiply (SI=MUL) Test (M8390 CPU Module) Rev 01.00 Page 188

:8570 : 7) Execute COND.ADD&SHIFT WR[2] to WR[0].Q. The MPLIER LSB H signal
:8571 : being high will cause an A PLUS B (80 00 00 01 plus 1). The shift
:8572 : will put a 1 in bit 0 and bit 30 of WR 0, and the Q reg will still
:8573 : be 0. Bit 31 of WR 0 will be 1 because N LONG is set and V LONG
:8574 : is clear. Check the results for these operands.
:8575 : 8) Repeat step 2 except load WR 2 and WR 0 with 40 00 00 00.
:8576 : 9) Execute COND.ADD&SHIFT WR[2] to WR[0].Q. The MPLIER LSB H signal
:8577 : being high will cause an A PLUS B (40 00 00 00 plus 40 00 00 00).
:8578 : The add will set bit 31 of WR 0 and the shift will put this 1 in
:8579 : bit 30 of WR 0. Bit 31 of WR 0 will be 0 because N LONG and V
:8580 : LONG are both set. The Q reg will still be 0. Check the results
:8581 : for these operands.
:8582 : 10) Repeat step 2 except load WR 2 and WR 0 with 80 00 00 00.
:8583 : 11) Execute COND.ADD&SHIFT WR[2] to WR[0].Q. The MPLIER LSB H signal
:8584 : being high will cause an A PLUS B (80 00 00 00 plus 80 00 00 00).
:8585 : The ADD will clear WR 0 and set V LONG. WR 0 will have a 1 in bit
:8586 : 31, and the Q reg will still be 0. Bit 31 of WR 0 will be 1 be-
:8587 : cause N LONG is clear and V LONG is set. Check the results for
:8588 : these operands.
:8589 : 12) Repeat step 2 except load WR 2 with a 2 and WR 0 with a 3.
:8590 : 13) Execute COND.SUB&SHIFT WR[2] FROM WR[0].Q. The MPLIER LSB H sig-
:8591 : nal being high will cause a B MINUS A (3 minus 2) to execute fol-
:8592 : lowed by a shift right. The shift will put a 1 in bit 31 of the
:8593 : Q reg and leave WR 0 clear. Bit 31 of WR 0 will be clear because
:8594 : both N LONG and V LONG are clear.
:8595 :
:8596 :
:8597 :
:8598 :
:8599 :
:8600 :
:8601 :
:8602 :
:8603 :
:8604 :
:8605 :
:8606 :
:8607 :
:8608 :
:8609 :
:8610 :
:8611 :
:8612 :
:8613 :
:8614 :
:8615 :
:8616 :
:8617 :
:8618 :
:8619 :
:8620 :
:8621 :
:8622 :
:8623 :
:8624 :

Errors:

Error 1 - WR 0 failed on COND.ADD&SHIFT WR[2] TO WR[0].Q with the
following data before add and shift: WR 2=1 WR 0=1 QREG=0
MPLIER=set. N LONG and V LONG will both be clear after add.
Error 2 - Same as error 1 except the Q reg failed.
Error 3 - WR 0 failed on COND.ADD&SHIFT WR[2] TO WR[0].Q with the
following data before add and shift: WR 2=1 WR 0=1 QREG=0
MPLIER=clear. N LONG and V LONG will both be clear after
add.
Error 4 - Same as error 3 except the Q reg failed.
Error 5 - WR 0 failed on COND.ADD&SHIFT WR[2] TO WR[0].Q with the
following data before add and shift: WR 2=80 00 00 00(H)
WR 0=1 QREG=0 MPLIER=set. N LONG will be set and V LONG
will be clear after add.
Error 6 - Same as error 5 except the Q reg failed.
Error 7 - WR 0 failed on COND.ADD&SHIFT WR[2] TO WR[0].Q with the
following data before add and shift: WR 2=40 00 00 00(H)
WR 0=40 00 00 00(H) QREG=0 MPLIER=set. N LONG and V LONG
will be set after add.
Error 8 - Same as error 7 except the Q reg failed.
Error 9 - WR 0 failed on COND.ADD&SHIFT WR[2] TO WR[0].Q with the
following data before add and shift: WR 2=80 00 00 00(H)
WR 0=80 00 00 00(H) QREG=0 MPLIER=set. N LONG will be
clear and V LONG will be set after add.
Error A- Same as error 9 except the Q reg failed.
Error B- WR 0 failed on COND.SUB&SHIFT WR[2] FROM WR[0].Q with the
following data before subtract and shift: WR 2=2 WR 0=3
QREG=0 MPLIER=set. N LONG and V LONG will be clear after

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

TEST C - Signed Mul 3-MAR-82 N 15
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module
TEST C - Signed Multiply (SI=MUL) Test (M8390 CPU Module)

Fiche 1 Frame N15

Sequence 195

Rev 01.00

Page 189

:8625 subtract.
:8626 Error C- Same as Error B except the Q reg failed.
:8627
:8628
:8629 Debug:
:8630 Error 1 - This error could result from several problems. First check
:8631 the control lines to the 2901 during the COND.ADD&SHIFT WR[2] TO
:8632 WR[0].Q instruction. The SRC should be A.B. Refer to the 2901
:8633 control line description at the beginning of this listing and this
:8634 test description for control signal values. If the control lines are
:8635 incorrect, check the inputs to the DEST CTL ROM and SRC.ALU CTL PAL
:8636 for a 0 1000 1000(B) on CSR bits 22-14. Also check MPLIER LSB H to
:8637 the SRC.ALU CTL PAL for a high. If these are correct, suspect the
:8638 DEST CTL ROM or SRC.ALU CTL PAL, depending on which output is bad.
:8639 If MPLIER LSB H is low, check the input Q SHF LSB H to the ALU N.Z PAL
:8640 during the ASHRC WR[3].Q instruction. This input should go high and
:8641 the output MPLIER LSB H should follow it. Q SHF LSB H has already
:8642 been checked, but the etch to this PAL could be open. If the input is
:8643 OK and the output is low, suspect the PAL itself. If the control
:8644 lines are OK, the error is probably in bit 31 of WR 0. Check the
:8645 signal RAM SHF MSB H during the COND.ADD&SHIFT WR[2] TO WR[0].Q for
:8646 a low at the SHIFT DATA PAL. If incorrect, check the inputs for a
:8647 low on N LONG H and V LONG H and check CSR bits 13-11 for a 4(H).
:8648 Suspect the SHIFT DATA PAL if these inputs are OK.
:8649
:8650 Error 2 - This error indicates a problem with the Q REG portion of
:8651 the COND.ADD&SHIFT WR[2] TO WR[0].Q instruction execution. First
:8652 check the control lines to the 2901 as in error 1. If these are
:8653 OK, the problem is probably in the SHIFT DATA PAL. Check the inputs
:8654 to this PAL during the conditional add instruction for a 4(H) on
:8655 CSR bit 13-11. If this is OK, the PAL is probably bad.
:8656
:8657 Error 3 - Same as error 1 with the following exceptions: MPLIER LSB
:8658 H will be low so the SRC mode should be 0.B. Q SHF LSB H will be low
:8659 at the ALU N.Z PAL producing the low on MPLIER LSB H.
:8660
:8661 Error 4 - Same as error 2 with the following exceptions: MPLIER LSB
:8662 H will be low so the SRC mode should be 0.B.
:8663
:8664 Error 5 - Same as error 1 with the following exceptions: RAM SHF MSB
:8665 H should be high, and N LONG H should be high at the SHIFT DATA PAL.
:8666
:8667 Error 6 - same as error 2.
:8668
:8669 Error 7 - Same as error 1 with the following exceptions: N LONG H and
:8670 V LONG H should both be high at the SHIFT DATA PAL. RAM SHF MSB H
:8671 should still be low as in error 1.
:8672
:8673 Error 8 - same as error 2.
:8674
:8675 Error 9 - Same as error 1 with the following exceptions: RAM SHF MSB
:8676 H should be high, and V LONG H should be high at the SHIFT DATA PAL.
:8677
:8678 Error A - same as error 2.
:8679

U 18F2, 0869,3C :8735

JSR [CHECK.RESULT]

ZZ-ENKCB-1.0 : ENKCB.MIC
: ENKCB.MIC
: ENKCB.MIC

TEST C - Signed Mul 3-MAR-82 B 16
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Rev 01.00 Sequence 196 Page 190
TEST C - Signed Multiply (SI=MUL) Test (M8390 CPU Module)

:8680 : Error B - This error most likely indicates a problem with the DEST
:8681 : CTL ROM or the SRC.ALU CTL PAL. The inputs on CSR 22-15 should be
:8682 : 0 1000 1001. Refer to the 2901 control line description and this
:8683 : test description for the COND.SUB&SHIFT WR[2] FROM WR[0] for the
:8684 : correct control line values.
:8685 :
:8686 : Error C - Same as Error B.
:8687 :
:8688 :
:8689 :
:8690 :
:8691 :
:8692 :
:8693 :
:8694 :
:8695 :
:8696 :
:8697 :
:8698 :
:8699 :
:8700 :
:8701 :
:8702 :
:8703 :
:8704 :
:8705 :
:8706 :
:8707 :
:8708 :
:8709 :
:8710 :
:8711 :
:8712 :
:8713 :
:8714 :
:8715 :
:8716 :
:8717 :
:8718 :
:8719 :
:8720 :
:8721 :
:8722 :
:8723 :
:8724 :
:8725 :
:8726 :
:8727 :
:8728 :
:8729 :
:8730 :
:8731 :
:8732 :
:8733 :
:8734 :

U 18CB, B65E,15 :8690

U 18CC, 3E80,15 :8691

U 18CD, 10E0,15 :8692

U 18CE, 098C,E4 :8693

U 18CF, E58A,15 :8694

U 18D0, 65A0,15 :8695

U 18D1, 3442,15 :8696

U 18D2, 3E8C,15 :8697

U 18D3, B440,15 :8698

U 18D4, BE82,15 :8699

U 18D5, B640,15 :8700

U 18D6, 2001,15 :8701

U 18D7, AAC0,15 :8702

U 18D8, 2F87,95 :8703

U 18D9, A319,95 :8704

U 18DA, 2224,15 :8705

U 18DB, 7610,15 :8706

U 18DC, 3640,95 :8707

U 18DD, 0869,3C :8708

U 18DE, 098D,34 :8709

U 18DF, 8870,0C :8710

U 18E0, B610,15 :8711

U 18E1, 2F82,95 :8712

U 18E2, 0869,3C :8713

U 18E3, 098D,34 :8714

U 18E4, 32C0,15 :8715

U 18E5, BE82,15 :8716

U 18E6, B640,15 :8717

U 18E7, 2001,15 :8718

U 18E8, AB80,15 :8719

U 18E9, A319,95 :8720

U 18EA, 2224,15 :8721

U 18EB, 7610,15 :8722

U 18EC, 2F82,95 :8723

U 18ED, 0869,3C :8724

U 18EE, 898E,44 :8725

U 18EF, 8870,0C :8726

U 18F0, B610,15 :8727

U 18F1, B67E,95 :8728

ZZ-ENKCB-1.0 : ANCCB\$MIC

: ENKCB.MIC

: ENKCB.MIC

T.C:

WAIT.TC.0:

TC.2:

TC.4:

MOV LS[BEGIN.TEST] TO WR[0] :SET BIT15 IN WRO FOR CONTROL AND STATUS WORD
MOV WR[0] TO LS[CONTROL.STATUS] :SET BIT15 IN CONTROL AND STATUS WORD
MISC [SET.CP.ATTN] :BIT15 INDICATES START OF TEST TO CONSOLE
ISSUE CPU ATTN TO CONSOLE

JMP [WAIT.TC.0] :LOOP HERE WAITING FOR CONSOLE TO RESPOND
CLR LS[ERRCR.MASK] :CLEAR ERROR MASK.
CLR LS[PREVIOUS.ERROR] :CLEAR PREVIOUS ERROR DUEBER

DSSCPUU TG UR[0]3QET BAT 1 GD WR 0
EGT WR[0] TE DS[EGDUDE&DUEU] 3SET UP EGDUE CEDE DOR CPU EGDUE
DOGP&TC&1:

EGV LS[#1] TO WR[0] :SET WRO TO 1
MOV WR[0] TO LS[ERROR.NUMBER] :SET ERROR NUMBER = 1
MOV LS[#1] TO WR[0] :WRO = 1
MOV WR[0] TO WR[2] :WR2 = 1
MOV WR[0] TO Q :Q = 1
CLR WR[3] :WR3 = 0
ASHRC WR[3],Q :LOAD MPLIER LSB H & Q = 0
COND.ADD&SHIFT WR[2] TO WR[0],Q
MOV Q TO LS[T8] :STORE Q
MOV LS[#1] TO WR[1] :EXPECTED (WRO) = 1
JSR [CHECK.RESULT]
JMP [LOOP.TC.1] :LOOP ON ERROR IF ENABLED
JSR [INC.EN] :INCREMENT ERROR NUMBER TO 2

MOV LS[T8] TO WR[0] :GET Q
CLR WR[1] :EXPECTED (Q) = 0
JSR [CHECK.RESULT]
JMP [LOOP.TC.1] :LOOP ON ERROR IF ENABLEDJABHED:831(HJCP*PC*3:

HSZ#3(H)X TK SRZOY
IKV SRZOY TK HS[ERRKR*JUIBERY] :SET ERRKR JUIBER 9 3

IKV HS[#1Y TK WR[0Y] :WRO = 1
MOV WR[0] TO WR[2] :WR2 = 1
CLR Q :Q = 0
ASHRC WR[3],Q :LOAD MPLIER LSB H
COND.ADD&SHIFT WR[2] TO WR[0],Q
MOV Q TO LS[T8] :STORE Q
CLR WR[1] :EXPECTED (WRO) = 0
JSR [CHECK.RESULT]
JEP [DOOP.TC.3] :LOOP ON ERROR IF ENABLED
JSR [IFC&EN] :IFCREMEFT ERRGR FUEBER TO 4

EGV DS[T8U TG WR[0] :EET Q
EGV DS[BIT31] TG WR[1U] :EXPECTED (Q) 5 20 00 00 00

TEST C - Signed Mul 3-MAR-82 B 16
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Rev 01.00 Sequence 197 Page 191
TEST C - Signed Multiply (SI=MUL) Test (M8390 CPU Module)

```

U 18F3, 898E,44 :8736      JMP [LOOP.TC.3]          ;LOOP ON ERROR IF ENABLED
U 18F4, 3644,15 :8737      LOOP.TC.5:
U 18F5, 2040,15 :8738      MOV LS[#4] TO WR[0]
U 18F6, BE82,15 :8739      INC WR[0]
U 18F7, B640,15 :8740      MOV WR[0] TO LS[ERROR.NUMBER] ;SET ERROR NUMBER = 5
U 18F8, 2001,15 :8741      MOV LS[#1] TO WR[0]          ;WRO = 1
U 18F9, C77F,15 :8742      MOV WR[0] TO WR[2]
U 18FA, AAC0,15 :8743      BIS LS[BIT31] TO WR[2]          ;WR2 = 80 00 00 01
U 18FB, A319,95 :8744      MOV WR[0] TO Q              ;Q = 1
U 18FC, 2224,15 :8745      ASHRC WR[3],Q                ;LOAD MPLIER LSB H
U 18FD, 7610,15 :8746      COND.ADD&SHIFT WR[2] TO WR[0],Q
U 18FE, B67E,95 :8747      MOV Q TO LS[T8]              ;STORE Q
U 18FF, 477C,95 :8748      MOV LS[BIT31] TO WR[1]
U 1900, 2042,95 :8749      BIS LS[BIT30] TO WR[1]
U 1901, 0869,3C :8750      INC WR[1]
U 1902, 098F,44 :8751      JSR [CHECK.RESULT]          ;EXPECTED (WRO) = C0 00 00 01
U 1903, 8870,0C :8752      JMP [LOOP.TC.5]              ;LOOP ON ERROR IF ENABLED
U 1904, B610,15 :8753      JSR [INC.EN]                ;INCREMENT ERROR NUMBER TO 6
U 1905, 2F82,95 :8754      TC.6:
U 1906, 0869,3C :8755      MOV LS[T8] TO WR[0]          ;GET Q
U 1907, 098F,44 :8756      CLR WR[1]                  ;EXPECTED (Q) = 0
U 1908, B646,15 :8757      JSR [CHECK.RESULT]
U 1909, C140,15 :8758      JMP [LOOP.TC.5]              ;LOOP ON ERROR IF ENABLED
U 190A, BE82,15 :8759      LOOP.TC.7:
U 190B, B67C,15 :8760      MOV LS[#8] TO WR[0]
U 190C, 2001,15 :8761      SUB LS[#1] FROM WR[0]
U 190D, 5840,15 :8762      MOV WR[0] TO LS[ERROR.NUMBER] ;SET ERROR NUMBER = 7
U 190E, A319,95 :8763      MOV LS[BIT30] TO WR[0]          ;WRO = 40 00 00 00
U 190F, 2224,15 :8764      MOV WR[0] TO WR[2]          ;WR2 = 40 00 00 00
U 1910, 7610,15 :8765      MOV LS[#1] TO Q              ;Q = 1
U 1911, 367C,95 :8766      ASHRC WR[3],Q                ;LOAD MPLIER LSB H
U 1912, 0869,3C :8767      COND.ADD&SHIFT WR[2] TO WR[0],Q
U 1913, 8990,84 :8768      MOV Q TO LS[T8]              ;STORE Q
U 1914, 8870,0C :8769      MOV LS[BIT30] TO WR[1]          ;EXPECTED (WRO) = 40 00 00 00
U 1915, B610,15 :8770      JSR [CHECK.RESULT]
U 1916, 2F82,95 :8771      JMP [LOOP.TC.7]              ;LOOP ON ERROR IF ENABLED
U 1917, 0869,3C :8772      JSR [INC.EN]                ;INCREMENT ERROR NUMBER TO 8
U 1918, 8990,84 :8773      TC.8:
U 1919, B646,15 :8774      MOV LS[T8] TO WR[0]          ;GET Q
U 191A, 2040,15 :8775      CLR WR[1]                  ;EXPECTED (Q) = 0
U 191B, BE82,15 :8776      JSR [CHECK.RESULT]
U 191C, 367E,15 :8777      JMP [LOOP.TC.7]              ;LOOP ON ERROR IF ENABLED
U 191D, 2001,15 :8778      LOOP.TC.9:
U 191E, 5840,15 :8779      MOV LS[#8] TO WR[0]
U 191F, A319,95 :8780      INC WR[0]
U 1920, 2224,15 :8781      MOV WR[0] TO LS[ERROR.NUMBER] ;SET ERROR NUMBER = 9
U 1921, 7610,15 :8782      MOV LS[BIT31] TO WR[0]          ;WRO = 80 00 00 00
U 1922, B67E,95 :8783      MOV WR[0] TO WR[2]          ;WR2 = 80 00 00 00
U 1923, 0869,3C :8784      MOV LS[#1] TO Q              ;Q = 1
U 1924, 2040,15 :8785      ASHRC WR[3],Q                ;LOAD MPLIER LSB H
U 1925, 2224,15 :8786      COND.ADD&SHIFT WR[2] TO WR[0],Q
U 1926, 7610,15 :8787      MOV Q TO LS[T8]              ;STORE Q
U 1927, B67E,95 :8788      MOV LS[BIT31] TO WR[1]          ;EXPECTED (WRO) = 80 00 00 00
U 1928, 0869,3C :8789      JSR [CHECK.RESULT]

```

```

U 1924, 8991,94 :8790      JMP [LOOP.TC.9]      ;LOOP ON ERROR IF ENABLED
U 1925, 8870,0C :8791      JSR [INC.EN]      ;INCREMENT ERROR NUMBER TO A
                                TC.A:
U 1926, 8610,15 :8792      MOV LS[T8] TO WR[0]      ;GET Q
U 1927, 2F82,95 :8793      CLR WR[1]      ;EXPECTED (Q) = 0
U 1928, 0869,3C :8794      JSR [CHECK.RESULT]
U 1929, 8991,94 :8795      JMP [LOOP.TC.9]      ;LOOP ON ERROR IF ENABLED
                                LOOP.TC.B:
U 192A, 8646,15 :8796      MOV LS[#8] TO WR[0]
U 192B, C0C0,15 :8797      ADD LS[#3(H)] TO WR[0]
U 192C, BE82,15 :8798      MOV WR[0] TO LS[ERROR.NUMBER] ;SET ERROR NUMBER = B
U 192D, 8643,15 :8800      MOV LS[#2] TO WR[2]      ;WR2 = 2
U 192E, 36C0,15 :8801      MOV LS[#3(H)] TO WR[0]      ;WR 0 = 3
U 192F, 5840,15 :8802      MOV LS[#1] TO Q      ;Q = 1
U 1930, A319,95 :8803      ASHRC WR[3],Q      ;LOAD MPLIER LSB H
U 1931, A264,15 :8804      COND.SUB&SHIFT WR[2] FROM WR[0],Q
U 1932, 7610,15 :8805      MOV Q TO LS[T8]      ;STORE Q
U 1933, 2F82,95 :8806      CLR WR[1]      ;EXPECTED (WR0) = 80 00 00 00
U 1934, 0869,3C :8807      JSR [CHECK.RESULT]
U 1935, 8992,A4 :8808      JMP [LOOP.TC.B]      ;LOOP ON ERROR IF ENABLED
U 1936, 8870,0C :8809      JSR [INC.EN]      ;INCREMENT ERROR NUMBER TO C
                                TC.C:
U 1937, 8610,15 :8810      MOV LS[T8] TO WR[0]      ;GET Q
U 1938, 867E,95 :8811      MOV LS[B1T31] TO WR[1]      ;EXPECTED (Q) = 0
U 1939, 0869,3C :8812      JSR [CHECK.RESULT]
U 193A, 8992,A4 :8813      JMP [LOOP.TC.B]      ;LOOP ON ERROR IF ENABLED
                                END.TC:
U 193B, 8870,0C :8814
U 193C, 8870,0C :8815
U 193D, 8870,0C :8816
  
```


:8817
:8818
:8819
:8820
:8821
:8822
:8823
:8824
:8825
:8826
:8827
:8828
:8829
:8830
:8831
:8832
:8833
:8834
:8835
:8836
:8837
:8838
:8839
:8840
:8841
:8842
:8843
:8844
:8845
:8846
:8847
:8848
:8849
:8850
:8851
:8852
:8853
:8854
:8855
:8856
:8857
:8858
:8859
:8860
:8861
:8862
:8863
:8864
:8865
:8866
:8867
:8868
:8869
:8870
:8871

Page "TEST D - Unsigned Multiply (SI=UMUL) Test (M8390 CPU Module)"

Test Description:

This test checks the unsigned multiply instruction and SHIFT DATA PAL. The instruction UNSIGNED.MUL WR TO WR.Q is used in this test. This instruction uses the same conditional add and shift mode as the signed multiply tested previously. The SRC mode is dependent on the value of the Q reg bit 0 before a previous shift operation. The control signals for the 2901 during the conditional instruction are as follows: SRC=MUL FUNC=R PLUS S DST=RSHF.RAM.Q CIN=NOCIN. the SRC mode will be either A.B, if the Q reg contained a 1 in bit 0 before a previous shift, or 0.B if the Q reg had a 0 in bit 0 before a previous shift. The SI field is set to a 5 for this test. Since the conditional add and shift was tested in the previous test, this test will only use the operands to make the SRC mode equal A.B. The test first loads 2 WR's and the Q reg with data and shifts the Q reg right to load MPLIER LSB H, which controls the SRC mode for the next instruction. then a UNSIGNED.MUL WR[2] TO WR[0].Q instruction is executed and the result is checked. Data patterns are picked that shift into the Q MSB.

Assumptions:

It is assumed that the previous signed multiply test has run successfully.

Test Steps:

- 1) Set up error mask, error number, and module number in LS (for error printouts) and clear previous error number in LS.
- 2) Load WR 2, WR 0, and the Q reg with a 1. Load WR 3 with all 0's, and execute ASHRC WR[3].Q. This loads MPLIER LSB H with a 1 and leaves 0 in the Q reg.
- 3) Execute UNSIGNED.MUL WR[2] to WR[0].Q. The MPLIER LSB H signal being high will cause an A PLUS B (1 plus 1). The shift will put a 1 in bit 0 of WR 0, and the Q reg will still be 0. Bit 31 of WR 0 will be 0 because CARRY 32 H is low. Check the results for these operands.
- 4) Repeat step 2 except load WR 2 with C0 00 00 01(H) and WR 0 with 40 00 00 00(H).
- 5) Execute UNSIGNED.MUL WR[2] to WR[0].Q. The MPLIER LSB H signal being high will cause an A PLUS B (C0 00 00 01 plus 60 00 00 00). The add will cause CARRY 32 H to be asserted so that bit 31 of WR 0 will be set. The Q reg will also get bit 31 set through the shift from WR 0 bit 0. Check the results for these operands.

Errors:

- Error 1 - WR 0 failed on UNSIGNED.MUL WR[2] TO WR[0].Q with the following data before add and shift: WR 2=1 WR 0=1 QREG=0 MPLIER=set. CARRY 32 H will be low after the add.
- Error 2 - Same as error 1 except the Q reg failed.

```

:8872      Error 3 - WR 0 failed on UNSIGNED.MUL WR[2] TO WR[0].Q with the
:8873      following data before add and shift: WR 2=C0 00 00 01(H)
:8874      WR 0=40 00 00 00(H) QREG=0 MPLIER=set. CARRY 32 H will
:8875      set after the add.
:8876      Error 4 - Same as error 5 except the Q reg failed.
:8877
:8878      Debug:
:8879
:8880      Error 1 - This error is probably in bit 31 of WR 0. Check the
:8881      signal RAM SHF MSB H during the UNSIGNED.MUL WR[2] TO WR[0].Q for
:8882      a low at the SHIFT DATA PAL. If incorrect, check the inputs for a
:8883      low on CARRY 32 H and check CSR bits 13-11 for a 5(H). Suspect the
:8884      SHIFT DATA PAL if these inputs are OK.
:8885
:8886      Error 2 - This error indicates a problem with the Q REG portion of
:8887      the UNSIGNED.MUL WR[2] TO WR[0].Q instruction execution. The problem
:8888      is probably in the SHIFT DATA PAL. Check the inputs to this PAL
:8889      during the conditional add instruction for a 5(H) on CSR bit 13-11.
:8890      If this is OK, the PAL is probably bad.
:8891
:8892      Error 3 - Same as error 1 with the following exceptions: RAM SHF MSB
:8893      H should be high, and CARRY 32 H should be high at the SHIFT DATA PAL.
:8894
:8895      Error 4 - Same as error 2.
:8896
:8897      --
:8898      T.D:
:8899      MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT15 IN WR0 FOR CONTROL AND STATUS WORD
:8900      MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT15 IN CONTROL AND STATUS WORD
:8901      ; BIT15 INDICATES START OF TEST TO CONSOLE
:8902      MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:8903      WAIT.TD.0:
:8904      JMP [WAIT.TD.0] ;LOOP HERE WAITING FOR CONSOLE TO RESPOND
:8905      CLR LS[ERROR.MASK] ;CLEAR ERROR MASK.
:8906      CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:8907      MOV LS[CPU] TO WR[0] ;SET BIT 1 IN WR
:8908      MOV WR[0] TO LS[MODULE.NUM] ;SET UP MODULE CODE FOR CPU MODULE
:8909      LOOP.TD.1:
:8910      MOV LS[#1] TO WR[0] ;SET WR 0 TO 1
:8911      MOV WR[0] TO LS[ERROR.NUMBER] ;SET ERROR NUMBER = 1
:8912      MOV LS[#1] TO WR[0] ;WR0 = 1
:8913      MOV WR[0] TO WR[2] ;WR2 = 1
:8914      MOV WR[0] TO Q ;Q = 1
:8915      CLR WR[3] ;WR3 = 0
:8916      ASHRC WR[3].Q ;LOAD MPLIER LSB H & Q = 0
:8917      UNSIGNED.MUL WR[2] TO WR[0].Q
:8918      MOV Q TO LS[T8] ;STORE Q
:8919      MOV LS[#1] TO WR[1] ;EXPECTED (WR0) = 1
:8920      JSR [CHECK.RESULT]
:8921      JMP [LOOP.TD.1] ;LOOP ON ERROR IF ENABLED
:8922      JSR [INC.EN] ;INCREMENT ERROR NUMBER TO 2
:8923
:8924      TD.2:
:8925      MOV LS[T8] TO WR[0] ;GET Q
:8926      CLR WR[1] ;EXPECTED (Q) = 0
:8927      JSR [CHECK.RESULT]
  
```

```

U 193B, B65E,15 :8899
U 193C, 3E80,15 :8900
:8901
U 193D, 10E0,15 :8902
:8903
U 193E, 8993,E4 :8904
U 193F, E58A,15 :8905
U 1940, 65A0,15 :8906
U 1941, 3642,15 :8907
U 1942, 3E8C,15 :8908
:8909
U 1943, B640,15 :8910
U 1944, BE82,15 :8911
U 1945, B640,15 :8912
U 1946, 2001,15 :8913
U 1947, AAC0,15 :8914
U 1948, 2F87,95 :8915
U 1949, A319,95 :8916
U 194A, A22C,15 :8917
U 194B, 7610,15 :8918
U 194C, 3640,95 :8919
U 194D, 0869,3C :8920
U 194E, 8994,34 :8921
U 194F, 8870,0C :8922
:8923
U 1950, B610,15 :8924
U 1951, 2F82,95 :8925
U 1952, 0869,3C :8926
  
```

U 1953, 8994,34	:8927	JMP [LOOP.TD.1]	;LOOP ON ERROR IF ENABLEDP
U 1954, 36C0,15	:8928	LOOP.TD.3:	
U 1955, BE82,15	:8929	MOV LS[#3(H)] TO WR[0]	
U 1956, B67C,15	:8930	MOV WR[0] TO LS[ERROR.NUMBER]	;SET ERROR NUMBER = 3
U 1957, 2001,15	:8931	MOV LS[BIT30] TO WR[0]	;WRO = 40 00 0C 00
U 1958, C77F,15	:8932	MOV WR[0] TO WR[2]	
U 1959, 2045,15	:8933	BIS LS[BIT31] TO WR[2]	
U 195A, 5840,15	:8934	INC WR[2]	;WR2 = C0 00 0C 01
U 195B, 2F87,95	:8935	MOV LS[#1] TO Q	; Q = 1
U 195C, A319,95	:8936	CLR WR[3]	;WR3 = 0
U 195D, A22C,15	:8937	ASHRC WR[3],Q	;LOAD MPLIER LSB H & Q = 0
U 195E, 7610,15	:8938	UNSIGNED.MUL WR[2] TO WR[0].Q	
U 195F, B67E,95	:8939	MOV Q TO LS[T8]	;STORE Q
U 1960, 0869,3C	:8940	MOV LS[BIT31] TO WR[1]	;EXPECTED (WRO) = 80 00 00 00
U 1961, 8995,44	:8941	JSR [CHECK.RESULT]	
U 1962, 8870,0C	:8942	JMP [LOOP.TD.3]	;LOOP ON ERROR IF ENABLED
	:8943	JSR [INC.EN]	;INCREMENT ERROR NUMBER TO 4
	:8944	TD.4:	
U 1963, B610,15	:8945	MOV LS[T8] TO WR[0]	;GET Q
U 1964, B67E,95	:8946	MOV LS[BIT31] TO WR[1]	;EXPECTED (Q) = 80 00 00 00
U 1965, 0869,3C	:8947	JSR [CHECK.RESULT]	
U 1966, 8995,44	:8948	JMP [LOOP.TD.3]	;LOOP ON ERROR IF ENABLED
	:8949	END.TD:	

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

TEST E - Sign Exten 3-MAR-82 H 16
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module
TEST E - Sign Extend Logic Test (M8390 CPU Module) Fiche 1 Frame H16
Sequence 202 Rev 01.00 Page 196

:8950 Page "TEST E - Sign Extend Logic Test (M8390 CPU Module)"
:8951 ++
:8952
:8953 Test Description:
:8954
:8955 This test checks a portion of the SIGN XTEND CTL PAL, part of the 2X4
:8956 MUX and the three 8 bit latches that do a sign extend on byte and word
:8957 data. Part of this logic can only be checked with memory data, and
:8958 will be tested in the memory controller micro diagnostics. This test
:8959 will load the OS reg with a data pattern of 0's with bit 7 either set
:8960 or clear. A MOVE instruction from LS[OS] to WR 0 will be performed
:8961 and the 32 bits will be checked. If OS 7 was a 0, all 32 bits will
:8962 be 0. If OS 7 was a 1, then bits 31-15 will be 1. This will check
:8963 the SIGN XTEND CTL PAL outputs 12 and 13, as well as the 2X4 MUX in-
:8964 put OS 7 and the outputs on pins 7 and 9.
:8965
:8966 Assumptions:
:8967
:8968 It assumed that previous OS tests have run successfully.
:8969
:8970 Test Steps:
:8971
:8972 1) Set up error mask, error number, and module number in LS (for
:8973 error printouts) and clear previous error number in LS.
:8974 2) Load OS with data of 0's. Execute MOVE from LS[OS] to WR 0.
:8975 Check result for all 0's.
:8976 3) Load OS with data of 80(H). Execute MOVE from LS[OS] to WR 0.
:8977 Check result for FF FF FF 80(H).
:8978
:8979 Errors:
:8980
:8981 Error 1 - Sign extend logic failed with positive data.
:8982 Error 2 - Sign extend logic failed with negative data.
:8983
:8984 Debug:
:8985
:8986 Error 1 - First check the enables on pin 1 of the three 8 bit latches
:8987 that output on the D bus during the MOVE LS[OS] to WR 0 instruction.
:8988 All three enables should be low. If not, check the SIGN XTEND CTL PAL
:8989 pins 12 and 13 for a low. If both are not low, check the inputs for
:8990 the following: CSR 21 H, CSR 17 H and CSR 18 H for a high, SEL REGS L
:8991 for a low, and DISABLE D-BUS H for a low. If these are correct, the
:8992 SIGN XTEND CTL PAL is probably bad. If the enables to the latches are
:8993 OK, then check the 2X4 MUX for lows on pins 7 and 9. If either output
:8994 is high, check for a low on input pins 4, 3, 12, 13, 1, and 15 and a
:8995 high on pin 2. If these are OK, the MUX is bad. Finally, if the
:8996 outputs are both low, check the inputs to the latches for a low,
:8997 especially if only one bit is failing. Suspect a bad latch if the
:8998 inputs are OK (make sure the pullup on pin 11 of each latch is high).
:8999
:9000 Error 2 - Same as error 1 except pins 4, 3, 12, 13, 9, and 7 of the
:9001 2X4 MUX should be high.
:9002
:9003
:9004 T.E:--

```

U 1967, B65E.15 :9005      MOV LS[BEGIN.TEST] TO WR[0]      ;SET BIT15 IN WRO FOR CONTROL AND STATUS WORD
U 1968, 3E80.15 :9006      MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT15 IN CNTROL AND STATUS WORD
                                :9007      : BIT15 INDICATES START OF TEST TO CONSOLE
U 1969, 10E0.15 :9008      MISC [SET.CP.ATTN]           ;ISSUE CPU ATTN TO CONSOLE
                                :9009
U 196A, 0996.A4 :9010      WAIT.TE.0:
U 196B, E58A.15 :9011      JMP [WAIT.TE.0]              ;LOOP HERE WAITING FOR CONSOLE TO RESPOND
U 196C, 65A0.15 :9012      CLR LS[ERROR.MASK]           ;CLEAR ERROR MASK.
U 196D, B640.15 :9013      CLR LS[PREVIOUS.ERROR]       ;CLEAR PREVIOUS ERROR NUMBER
U 196E, BE82.15 :9014      MOV LS[#1] TO WR[0]          ;SET WR 0 TO 1
U 196F, A3C0.15 :9015      MOV WR[0] TO LS[ERROR.NUMBER] ;SET ERROR NUMBER TO FIRST ERROR
U 1970, 3E8C.15 :9016      ROL WR[0]                   ;SET WRO TO 2
                                :9017      MOV WR[0] TO LS[MODULE.NUM] ;SET UP MODULE CODE FOR CPU MODULE
U 1971, E5F8.15 :9018      LOOP.TE.1:
U 1972, B6F8.15 :9019      CLR LS[OS]                   ;OS = 00 00 00 00
U 1973, 2F82.95 :9020      MOV LS[OS] TO WR[0]
U 1974, 0869.3C :9021      CLR WR[1]                   ;EXPECTED = 00 00 00 00
U 1975, 0997.14 :9022      JSR [CHECK.RESULT]
U 1976, 8870.0C :9023      JMP [LOOP.TE.1]              ;LOOP ON ERROR IF ENABLED
                                :9024      JSR [INC.EN]       ;INCREMENT ERROR NUMBER TO 2
U 1977, 364E.15 :9025      LOOP.TE.2:
U 1978, 3EF8.15 :9026      MOV LS[#80] TO WR[0]
U 1979, B6F8.15 :9027      MOV WR[0] TO LS[OS]          ;OS = 00 00 00 80
U 197A, 362C.95 :9028      MOV LS[OS] TO WR[0]
U 197B, C74E.95 :9029      MOV LS[FFFFFF00] TO WR[1]
U 197C, 0869.3C :9030      BIS LS[#80] TO WR[1]         ;EXPECTED = FF FF FF 80
U 197D, 0997.74 :9031      JSR [CHECK.RESULT]
                                :9032      JMP [LOOP.TE.2]    ;LOOP ON ERROR IF ENABLED
END.TE:
  
```

:9033 Page "TEST F - DEST CTL ROM And SRC.ALU CTL PAL Tests (M8390 CPU Module)"
 :9034 ++

:9035
 :9036 Test Description:
 :9037

:9038 This test will test every location in the DEST CTL ROM and all pos-
 :9039 sible inputs to the SRC.ALU CTL PAL. It is meant to verify that any
 :9040 u-code instruction will control the 2901 as expected, even if that in-
 :9041 struction has not been tested in the previous tests. Some instruc-
 :9042 tions already tested will not be repeated here. The test will be
 :9043 divided into sections corresponding to the type of instruction being
 :9044 tested i.e. BASIC, MOVE, EXTENDED, MEM.REQ, MISC, JUMP, and DECODE.
 :9045 The memory will be disabled throughout this test. Only the 2901 modes
 :9046 will be checked. It does not make sense to expand this description
 :9047 further. Instead refer to the comments in the listing.
 :9048

:9049 Assumptions:
 :9050

:9051 It is assumed that all previous tests have run successfully.
 :9052

:9053 Test Steps:
 :9054

:9055 See comments in listing.
 :9056

:9057 Errors:
 :9058

:9059 OTHER data : DEST CTL ROM Address
 :9060

:9061 Error 1 - JUMP micro instruction altered WR
 :9062 Error 2 - JUMP.OS micro instruction altered WR
 :9063 Error 3 - MISC micro instruction altered WR
 :9064 Error 4 - PORT micro instruction altered WR
 :9065 Error 5 - EXTENDED micro instruction error
 :9066 Error 6 - BASIC micro instruction error
 :9067 Error 7 - MOVE micro instruction error
 :9068

:9069 Debug:
 :9070

:9071 Any and all errors will involve either the DEST CTL ROM or the SRC.ALU
 :9072 CTL PAL. The expected outputs can be determined by referring to the
 :9073 2901 control lines table and the modes specified in the listing. In
 :9074 this manner, the failure can be traced to one of the two IC's listed
 :9075 above.
 :9076

:9077 --
 :9078

U 197E, B65E,15	:9079	MOV LS[BEGIN.TEST] TO WR[0]	:SET BIT15 IN WR0 FOR CONTROL AND STATUS WORD
U 197F, 3E80,15	:9080	MOV WR[0] TO LS[CONTROL.STATUS]	:SET BIT15 IN CNTRL AND STATUS WORD
	:9081		: BIT15 INDICATES START OF TEST TO CONSOLE
U 1980, 10E0,15	:9082	MISC [SET.CP.ATTN]	:ISSUE CPU ATTN TO CONSOLE
	:9083	WAIT.TF.0:	
U 1981, 0998,14	:9084	JMP [WAIT.TF.0]	:LOOP HERE WAITING FOR CONSOLE TO RESPOND
U 1982, E58A,15	:9085	CLR LS[ERROR.MASK]	:CLEAR ERROR MASK.
U 1983, 65A0,15	:9086	CLR LS[PREVIOUS.ERROR]	:CLEAR PREVIOUS ERROR NUMBER
U 1984, B640,15	:9087	MOV LS[#1] TO WR[0]	:SET WR 0 TO 1

```

U 1985, BE82,15 :9088      MOV WR[0] TO LS[ERROR.NUMBER] ;SET ERROR NUMBER TO FIRST ERROR
U 1986, A3C0,15 :9089      ROL WR[0] ;SET WRO TO 2
U 1987, 3E8C,15 :9090      MOV WR[0] TO LS[MODULE.NUM] ;SET UP MODULE CODE FOR CPU MODULE
U 1988, 8870,FC :9091      JSR [ASSERT.OTHER] ;'OTHER' DATA FIELD WILL BE USED
:9092
:9093      ;JUMP tests
:9094
:9095      tf.1:
U 1989, B699,95 :9096      MOV LS[ALTER.ODD] TO WR[3] ;SET UP PATTERN IN WR3
U 198A, B64A,15 :9097      MOV LS[#20] TO WR[0]
U 198B, BE88,15 :9098      MOV WR[0] TO LS[ADDRESS.DATA] ;INIT 'OTHER' DATA FIELD TO 20
:9099      LOOP.TF.1.20:
U 198C, 083F,FC :9100      JSR [RETURN.3FF] ;JUMP TO LOCATION & RETURN
U 198D, 8A6F,6C :9101      JSR [CHECK.WR3] ;CHECK THAT PATTERN IS STILL OK IN WR3
U 198E, 8998,C4 :9102      JMP [LOOP.TF.1.20] ;LOOP ON ERROR IF ENABLED
U 198F, 8A70,1C :9103      JSR [INC.OTHER] ;INCREMENT 'OTHER' DATA
:9104      LOOP.TF.1.21:
U 1990, 8871,8C :9105      JSR [RETURN.718] ;JUMP TO LOCATION & RETURN
U 1991, 8A6F,6C :9106      JSR [CHECK.WR3] ;CHECK THAT PATTERN IS STILL OK IN WR3
U 1992, 0999,04 :9107      JMP [LOOP.TF.1.21] ;LOOP ON ERROR IF ENABLED
U 1993, 8A70,1C :9108      JSR [INC.OTHER] ;INCREMENT 'OTHER' DATA
:9109      LOOP.TF.1.22:
U 1994, 8881,8C :9110      JSR [RETURN.818] ;JUMP TO LOCATION & RETURN
U 1995, 8A6F,6C :9111      JSR [CHECK.WR3] ;CHECK THAT PATTERN IS STILL OK IN WR3
U 1996, 8999,44 :9112      JMP [LOOP.TF.1.22] ;LOOP ON ERROR IF ENABLED
U 1997, 8A70,1C :9113      JSR [INC.OTHER] ;INCREMENT 'OTHER' DATA
:9114      LOOP.TF.1.23:
U 1998, 0881,9C :9115      JSR [RETURN.C18] ;JUMP TO LOCATION & RETURN
U 1999, 8A6F,6C :9116      JSR [CHECK.WR3] ;CHECK THAT PATTERN IS STILL OK IN WR3
U 199A, 8999,84 :9117      JMP [LOOP.TF.1.23] ;LOOP ON ERROR IF ENABLED
U 199B, 8A70,1C :9118      JSR [INC.OTHER] ;INCREMENT 'OTHER' DATA
:9119      LOOP.TF.1.24:
U 199C, 8901,8C :9120      JSR [RETURN.1018] ;JUMP TO LOCATION & RETURN
U 199D, 8A6F,6C :9121      JSR [CHECK.WR3] ;CHECK THAT PATTERN IS STILL OK IN WR3
U 199E, 0999,C4 :9122      JMP [LOOP.TF.1.24] ;LOOP ON ERROR IF ENABLED
U 199F, 8A70,1C :9123      JSR [INC.OTHER] ;INCREMENT 'OTHER' DATA
:9124      LOOP.TF.1.25:
U 19A0, 0941,8C :9125      JSR [RETURN.1418] ;JUMP TO LOCATION & RETURN
U 19A1, 8A6F,6C :9126      JSR [CHECK.WR3] ;CHECK THAT PATTERN IS STILL OK IN WR3
U 19A2, 099A,04 :9127      JMP [LOOP.TF.1.25] ;LOOP ON ERROR IF ENABLED
U 19A3, 8A70,1C :9128      JSR [INC.OTHER] ;INCREMENT 'OTHER' DATA
:9129      LOOP.TF.1.26:
U 19A4, 89A1,8C :9130      JSR [RETURN.1A18] ;JUMP TO LOCATION & RETURN
U 19A5, 8A6F,6C :9131      JSR [CHECK.WR3] ;CHECK THAT PATTERN IS STILL OK IN WR3
U 19A6, 899A,44 :9132      JMP [LOOP.TF.1.26] ;LOOP ON ERROR IF ENABLED
U 19A7, 8A70,1C :9133      JSR [INC.OTHER] ;INCREMENT 'OTHER' DATA
:9134      LOOP.TF.1.27:
U 19A8, 89C1,8C :9135      JSR [RETURN.1C18] ;JUMP TO LOCATION & RETURN
U 19A9, 8A6F,6C :9136      JSR [CHECK.WR3] ;CHECK THAT PATTERN IS STILL OK IN WR3
U 19AA, 899A,84 :9137      JMP [LOOP.TF.1.27] ;LOOP ON ERROR IF ENABLED
U 19AB, 8A70,1C :9138      JSR [INC.OTHER] ;INCREMENT 'OTHER' DATA
:9139      LOOP.TF.1.28:
U 19AC, 8A01,8C :9140      JSR [RETURN.2018] ;JUMP TO LOCATION & RETURN
U 19AD, 8A6F,6C :9141      JSR [CHECK.WR3] ;CHECK THAT PATTERN IS STILL OK IN WR3
U 19AE, 099A,C4 :9142      JMP [LOOP.TF.1.28] ;LOOP ON ERROR IF ENABLED
  
```

U 19AF, 8A70,1C	:9143	JSR [INC.OTHER]	: INCREMENT 'OTHER' DATA
U 19B0, 0A41,8C	:9144	LOOP.TF.1.29:	
U 19B1, 8A6F,6C	:9145	JSR [RETURN.2418]	: JUMP TO LOCATION & RETURN
U 19B2, 899B,04	:9146	JSR [CHECK.WR3]	: CHECK THAT PATTERN IS STILL OK IN WR3
U 19B3, 8A70,1C	:9147	JMP [LOOP.TF.1.29]	: LOOP ON ERROR IF ENABLED
	:9148	JSR [INC.OTHER]	: INCREMENT 'OTHER' DATA
	:9149	LOOP.TF.1.2A:	
U 19B4, 0A81,8C	:9150	JSR [RETURN.2818]	: JUMP TO LOCATION & RETURN
U 19B5, 8A6F,6C	:9151	JSR [CHECK.WR3]	: CHECK THAT PATTERN IS STILL OK IN WR3
U 19B6, 099B,44	:9152	JMP [LOOP.TF.1.2A]	: LOOP ON ERROR IF ENABLED
U 19B7, 8A70,1C	:9153	JSR [INC.OTHER]	: INCREMENT 'OTHER' DATA
	:9154	LOOP.TF.1.2B:	
U 19B8, 8AC1,8C	:9155	JSR [RETURN.2C18]	: JUMP TO LOCATION & RETURN
U 19B9, 8A6F,6C	:9156	JSR [CHECK.WR3]	: CHECK THAT PATTERN IS STILL OK IN WR3
U 19BA, 099B,84	:9157	JMP [LOOP.TF.1.2B]	: LOOP ON ERROR IF ENABLED
U 19BB, 8A70,1C	:9158	JSR [INC.OTHER]	: INCREMENT 'OTHER' DATA
	:9159	LOOP.TF.1.2C:	
U 19BC, 0B01,8C	:9160	JSR [RETURN.3018]	: JUMP TO LOCATION & RETURN
U 19BD, 8A6F,6C	:9161	JSR [CHECK.WR3]	: CHECK THAT PATTERN IS STILL OK IN WR3
U 19BE, 899B,C4	:9162	JMP [LOOP.TF.1.2C]	: LOOP ON ERROR IF ENABLED
U 19BF, 8A70,1C	:9163	JSR [INC.OTHER]	: INCREMENT 'OTHER' DATA
	:9164	LOOP.TF.1.2D:	
U 19C0, 8B41,8C	:9165	JSR [RETURN.3418]	: JUMP TO LOCATION & RETURN
U 19C1, 8A6F,6C	:9166	JSR [CHECK.WR3]	: CHECK THAT PATTERN IS STILL OK IN WR3
U 19C2, 099C,04	:9167	JMP [LOOP.TF.1.2D]	: LOOP ON ERROR IF ENABLED
U 19C3, 8A70,1C	:9168	JSR [INC.OTHER]	: INCREMENT 'OTHER' DATA
	:9169	LOOP.TF.1.2E:	
U 19C4, 8B81,8C	:9170	JSR [RETURN.3818]	: JUMP TO LOCATION & RETURN
U 19C5, 8A6F,6C	:9171	JSR [CHECK.WR3]	: CHECK THAT PATTERN IS STILL OK IN WR3
U 19C6, 899C,44	:9172	JMP [LOOP.TF.1.2E]	: LOOP ON ERROR IF ENABLED
U 19C7, 8A70,1C	:9173	JSR [INC.OTHER]	: INCREMENT 'OTHER' DATA
	:9174	LOOP.TF.1.2F:	
U 19C8, 0BC1,8C	:9175	JSR [RETURN.3C18]	: JUMP TO LOCATION & RETURN
U 19C9, 8A6F,6C	:9176	JSR [CHECK.WR3]	: CHECK THAT PATTERN IS STILL OK IN WR3
U 19CA, 899C,84	:9177	JMP [LOOP.TF.1.2F]	: LOOP ON ERROR IF ENABLED
	:9178	TF.2:	
U 19CB, 8870,0C	:9179	JSR [INC.EN]	: INCREMENT ERROR NUMBER TO 2
U 19CC, E5F8,15	:9180	CLR LS[OS]	: INSURE THAT OS IS CLEAR
U 19CD, B64A,15	:9181	MOV LS[#20] TO WR[0]	
U 19CE, C048,15	:9182	ADD LS[#10] TO WR[0]	
U 19CF, BE88,15	:9183	MOV WR[0] TO LS[ADDRESS.DATA]	: INIT 'OTHER' DATA FIELD TO 30
	:9184	LOOP.TF.2.30:	
U 19D0, 8C3F,FC	:9185	JSR.OS [RETURN.3FF]	: JUMP TO LOCATION & RETURN
U 19D1, 8A6F,6C	:9186	JSR [CHECK.WR3]	: CHECK THAT PATTERN IS STILL OK IN WR3
U 19D2, 899D,04	:9187	JMP [LOOP.TF.2.30]	: LOOP ON ERROR IF ENABLED
U 19D3, 8A70,1C	:9188	JSR [INC.OTHER]	: INCREMENT 'OTHER' DATA
	:9189	LOOP.TF.2.31:	
U 19D4, 0C71,8C	:9190	JSR.OS [RETURN.718]	: JUMP TO LOCATION & RETURN
U 19D5, 8A6F,6C	:9191	JSR [CHECK.WR3]	: CHECK THAT PATTERN IS STILL OK IN WR3
U 19D6, 099D,44	:9192	JMP [LOOP.TF.2.31]	: LOOP ON ERROR IF ENABLED
U 19D7, 8A70,1C	:9193	JSR [INC.OTHER]	: INCREMENT 'OTHER' DATA
	:9194	LOOP.TF.2.32:	
U 19D8, 0C81,8C	:9195	JSR.OS [RETURN.818]	: JUMP TO LOCATION & RETURN
U 19D9, 8A6F,6C	:9196	JSR [CHECK.WR3]	: CHECK THAT PATTERN IS STILL OK IN WR3
U 19DA, 099D,84	:9197	JMP [LOOP.TF.2.32]	: LOOP ON ERROR IF ENABLED

U 19DB, 8A70,1C	:9198	JSR [INC.OTHER]	:INCREMENT "OTHER" DATA
	:9199	LOOP.TF.2.33:	
U 19DC, 8C81,9C	:9200	JSR.OS [RETURN,C18]	:JUMP TO LOCATION & RETURN
U 19DD, 8A6F,6C	:9201	JSR [CHECK.WR3]	:CHECK THAT PATTERN IS STILL OK IN WR3
U 19DE, 899D,C4	:9202	JMP [LOOP.TF.2.33]	:LOOP ON ERROR IF ENABLED
U 19DF, 8A70,1C	:9203	JSR [INC.OTHER]	:INCREMENT "OTHER" DATA
	:9204	LOOP.TF.2.34:	
U 19E0, 0D01,8C	:9205	JSR.OS [RETURN,1018]	:JUMP TO LOCATION & RETURN
U 19E1, 8A6F,6C	:9206	JSR [CHECK.WR3]	:CHECK THAT PATTERN IS STILL OK IN WR3
U 19E2, 899E,04	:9207	JMP [LOOP.TF.2.34]	:LOOP ON ERROR IF ENABLED
U 19E3, 8A70,1C	:9208	JSR [INC.OTHER]	:INCREMENT "OTHER" DATA
	:9209	LOOP.TF.2.35:	
U 19E4, 8D41,8C	:9210	JSR.OS [RETURN,1418]	:JUMP TO LOCATION & RETURN
U 19E5, 8A6F,6C	:9211	JSR [CHECK.WR3]	:CHECK THAT PATTERN IS STILL OK IN WR3
U 19E6, 099E,44	:9212	JMP [LOOP.TF.2.35]	:LOOP ON ERROR IF ENABLED
U 19E7, 8A70,1C	:9213	JSR [INC.OTHER]	:INCREMENT "OTHER" DATA
	:9214	LOOP.TF.2.36:	
U 19E8, 0DA1,8C	:9215	JSR.OS [RETURN,1A18]	:JUMP TO LOCATION & RETURN
U 19E9, 8A6F,6C	:9216	JSR [CHECK.WR3]	:CHECK THAT PATTERN IS STILL OK IN WR3
U 19EA, 099E,84	:9217	JMP [LOOP.TF.2.36]	:LOOP ON ERROR IF ENABLED
U 19EB, 8A70,1C	:9218	JSR [INC.OTHER]	:INCREMENT "OTHER" DATA
	:9219	LOOP.TF.2.37:	
U 19EC, 0DC1,8C	:9220	JSR.OS [RETURN,1C18]	:JUMP TO LOCATION & RETURN
U 19ED, 8A6F,6C	:9221	JSR [CHECK.WR3]	:CHECK THAT PATTERN IS STILL OK IN WR3
U 19EE, 899E,C4	:9222	JMP [LOOP.TF.2.37]	:LOOP ON ERROR IF ENABLED
U 19EF, 8A70,1C	:9223	JSR [INC.OTHER]	:INCREMENT "OTHER" DATA
	:9224	LOOP.TF.2.38:	
U 19F0, 0E01,8C	:9225	JSR.OS [RETURN,2018]	:JUMP TO LOCATION & RETURN
U 19F1, 8A6F,6C	:9226	JSR [CHECK.WR3]	:CHECK THAT PATTERN IS STILL OK IN WR3
U 19F2, 099F,04	:9227	JMP [LOOP.TF.2.38]	:LOOP ON ERROR IF ENABLED
U 19F3, 8A70,1C	:9228	JSR [INC.OTHER]	:INCREMENT "OTHER" DATA
	:9229	LOOP.TF.2.39:	
U 19F4, 8E41,8C	:9230	JSR.OS [RETURN,2418]	:JUMP TO LOCATION & RETURN
U 19F5, 8A6F,6C	:9231	JSR [CHECK.WR3]	:CHECK THAT PATTERN IS STILL OK IN WR3
U 19F6, 899F,44	:9232	JMP [LOOP.TF.2.39]	:LOOP ON ERROR IF ENABLED
U 19F7, 8A70,1C	:9233	JSR [INC.OTHER]	:INCREMENT "OTHER" DATA
	:9234	LOOP.TF.2.3A:	
U 19F8, 8E81,8C	:9235	JSR.OS [RETURN,2818]	:JUMP TO LOCATION & RETURN
U 19F9, 8A6F,6C	:9236	JSR [CHECK.WR3]	:CHECK THAT PATTERN IS STILL OK IN WR3
U 19FA, 899F,84	:9237	JMP [LOOP.TF.2.3A]	:LOOP ON ERROR IF ENABLED
U 19FB, 8A70,1C	:9238	JSR [INC.OTHER]	:INCREMENT "OTHER" DATA
	:9239	LOC.TF.2.3B:	
U 19FC, 0EC1,8C	:9240	JSR.OS [RETURN,2C18]	:JUMP TO LOCATION & RETURN
U 19FD, 8A6F,6C	:9241	JSR [CHECK.WR3]	:CHECK THAT PATTERN IS STILL OK IN WR3
U 19FE, 099F,C4	:9242	JMP [LOOP.TF.2.3B]	:LOOP ON ERROR IF ENABLED
U 19FF, 8A70,1C	:9243	JSR [INC.OTHER]	:INCREMENT "OTHER" DATA
	:9244	LOOP.TF.2.3C:	
U 1A00, 8F01,8C	:9245	JSR.OS [RETURN,3018]	:JUMP TO LOCATION & RETURN
U 1A01, 8A6F,6C	:9246	JSR [CHECK.WR3]	:CHECK THAT PATTERN IS STILL OK IN WR3
U 1A02, 09A0,04	:9247	JMP [LOOP.TF.2.3C]	:LOOP ON ERROR IF ENABLED
U 1A03, 8A70,1C	:9248	JSR [INC.OTHER]	:INCREMENT "OTHER" DATA
	:9249	LOOP.TF.2.3D:	
U 1A04, 0F41,8C	:9250	JSR.OS [RETURN,3418]	:JUMP TO LOCATION & RETURN
U 1A05, 8A6F,6C	:9251	JSR [CHECK.WR3]	:CHECK THAT PATTERN IS STILL OK IN WR3
U 1A06, 89A0,44	:9252	JMP [LOOP.TF.2.3D]	:LOOP ON ERROR IF ENABLED

```

U 1A07, 8A70,1C :9253      JSR [INC.OTHER]          ;INCREMENT "OTHER" DATA
:9254      LOOP.TF.2.3E:
U 1A08, 0F81,8C :9255      JSR.OS [RETURN,3818]        ;JUMP TO LOCATION & RETURN
U 1A09, 8A6F,6C :9256      JSR [CHECK.WR3]          ;CHECK THAT PATTERN IS STILL OK IN WR3
U 1A0A, 89A0,84 :9257      JMP [LOOP.TF.2.3E]        ;LOOP ON ERROR IF ENABLED
U 1A0B, 8A70,1C :9258      JSR [INC.OTHER]          ;INCREMENT "OTHER" DATA
:9259      LOOP.TF.2.3F:
U 1A0C, 8FC1,8C :9260      JSR.OS [RETURN,3C18]        ;JUMP TO LOCATION & RETURN
U 1A0D, 8A6F,6C :9261      JSR [CHECK.WR3]          ;CHECK THAT PATTERN IS STILL OK IN WR3
U 1A0E, 09A0,C4 :9262      JMP [LOOP.TF.2.3F]        ;LOOP ON ERROR IF ENABLED
U 1A0F, 89A1,A4 :9263      JMP [L.1A1A]
:9264      ;
:9265      ;MISC instruction
:9266      ;
:9267      1A1A:
:9268      L.1A1A:
:9269      TF.3:
U 1A1A, 8870,0C :9270      JSR [INC.EN]              ;INCREMENT ERROR NUMBER TO 3
U 1A1B, B64C,15 :9271      MOV LS[#40] TO WR[0]
U 1A1C, 2100,15 :9272      DEC WR[0]
U 1A1D, BE88,15 :9273      MOV WR[0] TO LS[ADDRESS.DATA] ;INIT "OTHER" DATA TO 3F
U 1A1E, 369A,15 :9274      MOV LS[ALTER.EVEN] TO WR[0] ;WRITE PATTERN IN WR 0
:9275      LOOP.TF.3.3F:
U 1A1F, 9030,15 :9276      MISC [CLR.S0]
U 1A20, 8A6F,CC :9277      JSR [CHECK.WR0]          ;CHECK THAT MISC INS. DID NOT ALTER WRO
U 1A21, 89A1,F4 :9278      JMP [LOOP.TF.3.3F]        ;LOOP ON ERROR IF ENABLED
U 1A22, 8A70,1C :9279      JSR [INC.OTHER]          ;INCREMENT "OTHER" DATA
:9280      LOOP.TF.3.40:
U 1A23, 9050,15 :9281      MISC [SET.S0]
U 1A24, 8A6F,CC :9282      JSR [CHECK.WR0]          ;CHECK THAT MISC INS. DID NOT ALTER WRO
U 1A25, 89A2,34 :9283      JMP [LOOP.TF.3.40]        ;LOOP ON ERROR IF ENABLED
U 1A26, 8A70,1C :9284      JSR [INC.OTHER]          ;INCREMENT "OTHER" DATA
:9285      LOOP.TF.3.41:
U 1A27, 1080,15 :9286      MISC [SET.ACC.1.0]
U 1A28, 8A6F,CC :9287      JSR [CHECK.WR0]          ;CHECK THAT MISC INS. DID NOT ALTER WRO
U 1A29, 09A2,74 :9288      JMP [LOOP.TF.3.41]        ;LOOP ON ERROR IF ENABLED
U 1A2A, 8A70,1C :9289      JSR [INC.OTHER]          ;INCREMENT "OTHER" DATA
:9290      LOOP.TF.3.42:
U 1A2B, 90F0,15 :9291      MISC [NOP]
U 1A2C, 8A6F,CC :9292      JSR [CHECK.WR0]          ;CHECK THAT MISC INS. DID NOT ALTER WRO
U 1A2D, 09A2,B4 :9293      JMP [LOOP.TF.3.42]        ;LOOP ON ERROR IF ENABLED
:9294      ;
:9295      TF.4:
U 1A2E, 8870,0C :9296      JSR [INC.EN]              ;INCREMENT ERROR NUMBER TO 4
U 1A2F, B64C,15 :9297      MOV LS[#40] TO WR[0]
U 1A30, 4748,15 :9298      BIS LS[#10] TO WR[0]
U 1A31, 2100,15 :9299      DEC WR[0]
U 1A32, BE88,15 :9300      MOV WR[0] TO LS[ADDRESS.DATA] ;INIT "OTHER" DATA TO 4F
U 1A33, 369A,15 :9301      MOV LS[ALTER.EVEN] TO WR[0]
:9302      LOOP.TF.4.4F:
U 1A34, 1400,15 :9303      OPC2/MISC,MISC.PORT/PORT,R.W.FUNC/O,PORT.SELECT/O,PORT.FUNC/READ
U 1A35, 8A6F,CC :9304      JSR [CHECK.WR0]          ;CHECK THAT PORT INS. DID NOT ALTER WRO
U 1A36, 89A3,44 :9305      JMP [LOOP.TF.4.4F]        ;LOOP ON ERROR IF ENABLED
U 1A37, 8A70,1C :9306      JSR [INC.OTHER]          ;INCREMENT "OTHER" DATA
:9307      LOOP.TF.4.50:
  
```



```

U 1A64, 1700,15 :9363      OPC2/MISC,MISC.PORT/PORT,R.W.FUNC/0,PORT.SELECT/6,PORT.FUNC/READ
U 1A65, 8A6F,CC :9364      JSR [CHECK.WR0]          ;CHECK THAT PORT INS. DID NOT ALTER WRO
U 1A66, 89A6,44 :9365      JMP [LOOP.TF.4.5B]        ;LOOP ON ERROR IF ENABLED
U 1A67, 8A70,1C :9366      JSR [INC.OTHER]          ;INCREMENT "OTHER" DATA
:9367      LOOP.TF.4.5C:
U 1A68, 9740,15 :9368      OPC2/MISC,MISC.PORT/PORT,R.W.FUNC/0,PORT.SELECT/6,PORT.FUNC/CONTROL
U 1A69, 8A6F,CC :9369      JSR [CHECK.WR0]          ;CHECK THAT PORT INS. DID NOT ALTER WRO
U 1A6A, 89A6,84 :9370      JMP [LOOP.TF.4.5C]        ;LOOP ON ERROR IF ENABLED
U 1A6B, 8A70,1C :9371      JSR [INC.OTHER]          ;INCREMENT "OTHER" DATA
:9372      LOOP.TF.4.5D:
U 1A6C, 9780,15 :9373      OPC2/MISC,MISC.PORT/PORT,R.W.FUNC/0,PORT.SELECT/7,PORT.FUNC/READ
U 1A6D, 8A6F,CC :9374      JSR [CHECK.WR0]          ;CHECK THAT PORT INS. DID NOT ALTER WRO
U 1A6E, 09A6,C4 :9375      JMP [LOOP.TF.4.5D]        ;LOOP ON ERROR IF ENABLED
U 1A6F, 8A70,1C :9376      JSR [INC.OTHER]          ;INCREMENT "OTHER" DATA
:9377      LOOP.TF.4.5E:
U 1A70, 17C0,15 :9378      OPC2/MISC,MISC.PORT/PORT,R.W.FUNC/0,PORT.SELECT/7,PORT.FUNC/CONTROL
U 1A71, 8A6F,CC :9379      JSR [CHECK.WR0]          ;CHECK THAT PORT INS. DID NOT ALTER WRO
U 1A72, 89A7,04 :9380      JMP [LOOP.TF.4.5E]        ;LOOP ON ERROR IF ENABLED
:9381      ;
:9382      ;EXTENDED Instruction Tests
:9383      ;
:9384      ;tf.5:
U 1A73, 8870,0C :9385      JSR [INC.EN]              ;INCREMENT ERROR NUMBER TO 5
U 1A74, 364E,15 :9386      MOV LS[#80] TO WR[0]
U 1A75, BE88,15 :9387      MOV WR[0] TO LS[ADDRESS.DATA] ;INIT "OTHER" DATA TO 80
:9388      ;
:9389      ;LOOP.TF.5.80:
U 1A76, 2F80,15 :9389      CLR WR[0]
U 1A77, 3698,95 :9390      MOV LS[ALTER.ODD] TO WR[1] ;SET UP DATA
U 1A78, 2002,15 :9391      MOV WR[1] TO WR[0]
U 1A79, 0869,3C :9392      JSR [CHECK.RESULT]        ;CHECK WR.PLUS.0.TO.WR
U 1A7A, 89A7,64 :9393      JMP [LOOP.TF.5.80]        ;LOOP ON ERROR IF ENABLED
U 1A7B, 8A70,1C :9394      JSR [INC.OTHER]          ;WR.INC.WR PREVIOUSLY TESTED
U 1A7C, 8A70,1C :9395      JSR [INC.OTHER]          ;INC "OTHER" DATA TO 82
:9396      ;
:9397      ;LOOP.TF.5.82:
U 1A7D, B640,15 :9397      MOV LS[#1] TO WR[0]          ;DATA = 1
U 1A7E, 369E,95 :9398      MOV LS[ONES] TO WR[1]        ;EXPECTED = -1
U 1A7F, 2080,15 :9399      NEG WR[0]
U 1A80, 0869,3C :9400      JSR [CHECK.RESULT]        ;CHECK WR.NEG.WR
U 1A81, 09A7,D4 :9401      JMP [LOOP.TF.5.82]        ;LOOP ON ERROR IF ENABLED
U 1A82, 8A70,1C :9402      JSR [INC.OTHER]          ;INC "OTHER" DATA TO 83
:9403      ;
:9404      ;LOOP.TF.5.83:
U 1A83, B698,15 :9404      MOV LS[ALTER.ODD] TO WR[0]        ;DATA = AAAAAAAA
U 1A84, B69A,95 :9405      MOV LS[ALTER.EVEN] TO WR[1]      ;EXPECTED = 55555555
U 1A85, A0C0,15 :9406      COM WR[0]
U 1A86, 0869,3C :9407      JSR [CHECK.RESULT]        ;CHECK WR.COM.WR
U 1A87, 89A8,34 :9408      JMP [LOOP.TF.5.83]        ;LOOP ON ERROR IF ENABLED
U 1A88, 8A70,1C :9409      JSR [INC.OTHER]          ;WR.DEC.WR PREVIOUSLY TESTED
U 1A89, 8A70,1C :9410      JSR [INC.OTHER]          ;85 EMPTY
U 1A8A, 8A70,1C :9411      JSR [INC.OTHER]          ;INC "OTHER" DATA TO 86
:9412      ;
:9413      ;LOOP.TF.5.86:
U 1A8B, 5898,15 :9413      MOV LS[ALTER.ODD] TO Q          ;DATA = AAAAAAAA
U 1A8C, 3698,95 :9414      MOV LS[ALTER.ODD] TO WR[1]      ;EXPECTED = AAAAAAAA
U 1A8D, A180,15 :9415      MOV Q TO WR[0]
U 1A8E, 0869,3C :9416      JSR [CHECK.RESULT]        ;CHECK MOV.Q.WR
U 1A8F, 09A8,B4 :9417      JMP [LOOP.TF.5.86]        ;LOOP ON ERROR IF ENABLED
  
```

U 1A90, 8A70.1C :9418	JSR [INC.OTHER]	:87 EMPTY
U 1A91, 8A70.1C :9419	JSR [INC.OTHER]	:COND.ADD&SHIFT PREVIOUSLY TESTED
U 1A92, 8A70.1C :9420	JSR [INC.OTHER]	:COND.SUB&SHIFT PREVIOUSLY TESTED
U 1A93, 8A70.1C :9421	JSR [INC.OTHER]	:8A EMPTY
U 1A94, 8A70.1C :9422	JSR [INC.OTHER]	:INC 'OTHER' DATA TO 8B
U 1A95, 8A70.1C :9423	JSR [INC.OTHER]	
U 1A95, B640.15 :9424	MOV LS[BIT0] TO WR[0]	:DATA = 00000001
U 1A96, B644.95 :9425	MOV LS[BIT2] TO WR[1]	:EXPECTED = 00000004
U 1A97, A2D0.15 :9426	SHL2 WR[0]	
U 1A98, 0869.3C :9427	JSR [CHECK.RESULT]	:CHECK SHL2
U 1A99, 09A9.54 :9428	JMP [LOOP.TF.5.88]	:LOOP ON ERROR IF ENABLED
U 1A9A, 8A70.1C :9429	JSR [INC.OTHER]	:ASHRC ALREADY TESTED
U 1A9B, 8A70.1C :9430	JSR [INC.OTHER]	:ASHR ALREADY TESTED
U 1A9C, 8A70.1C :9431	JSR [INC.OTHER]	:ASHLC ALREADY TESTED
U 1A9D, 8A70.1C :9432	JSR [INC.OTHER]	:ASHL ALREADY TESTED
U 1A9E, 8A70.1C :9433	JSR [INC.OTHER]	:INC 'OTHER' DATA TO 90
U 1A9F, DF46.15 :9434	JSR [INC.OTHER]	
U 1A9F, DF46.15 :9435	LOOP.TF.5.90:	
U 1AA0, 3E8A.15 :9436	MCOM LS[BIT3] TO WR[0]	
U 1AA1, 589E.15 :9437	MOV WR[0] TO LS[ERROR.MASK]	:MASK ALL BUT BIT #3 (TO LOOK AT N BIT)
U 1AA2, 3646.95 :9438	MOV LS[ONES] TO Q	:DATA = -1
U 1AA3, A400.35 :9439	MOV LS[BIT3] TO WR[1]	:EXPECTED = ALU N SET
U 1AA4, 7610.15 :9440	TST Q,	
U 1AA5, B7FC.15 :9441	DT(LONG)&SET.ALU.CC	
U 1AA6, 0869.3C :9442	MOV Q TO LS[T8]	
U 1AA7, 09A9.F4 :9443	MOV LS[ALU.CC] TO WR[0]	
U 1AA8, E58A.15 :9444	JSR [CHECK.RESULT]	:CHECK FOR N SET
U 1AA9, 369E.95 :9445	JMP [LOOP.TF.5.90]	:LOOP ON ERROR IF ENABLED
U 1AAA, B610.15 :9446	CLR LS[ERROR.MASK]	
U 1AAB, 0869.3C :9447	MOV LS[ONES] TO WR[1]	
U 1AAC, 09A9.F4 :9448	MOV LS[T8] TO WR[0]	:RETRIEVE Q
U 1AAD, 8A70.1C :9449	JSR [CHECK.RESULT]	:CHECK FOR Q NOT WRITTEN
U 1AAE, 8A70.1C :9450	JMP [LOOP.TF.5.90]	:LOOP ON ERROR IF ENABLED
U 1AAF, 2F80.15 :9451	JSR [INC.OTHER]	:91 EMPTY
U 1AB0, 5840.15 :9452	JSR [INC.OTHER]	:INC 'OTHER' DATA TO 92
U 1AB1, 2480.35 :9453	LOOP.TF.5.92:	
U 1AB2, 7610.15 :9454	CLR WR[0]	:WRO = 0
U 1AB3, 37FC.95 :9455	MOV LS[#1] TO Q	:Q = 1
U 1AB4, 3E12.95 :9456	CMP WR[0] WITH Q,	
U 1AB5, 2F82.95 :9457	DT(LONG)&SET.ALU.CC	
U 1AB6, 0869.3C :9458	MOV Q TO LS[T8]	:SAVE Q
U 1AB7, 09AA.F4 :9459	MOV LS[ALU.CC] TO WR[1]	
U 1AB8, DF46.15 :9460	MOV WR[1] TO LS[T9]	
U 1AB9, 3E8A.15 :9461	CLR WR[1]	
U 1ABA, 3646.95 :9462	JSR [CHECK.RESULT]	:CHECK WRO NOT WRITTEN
U 1ABB, 3612.15 :9463	JMP [LOOP.TF.5.92]	:LOOP ON ERROR IF ENABLED
U 1ABC, 0869.3C :9464	MCOM LS[BIT3] TO WR[0]	
U 1ABD, 09AA.F4 :9465	MOV WR[0] TO LS[ERROR.MASK]	:MASK ALL BUT BIT #3 (TO LOOK AT N BIT)
U 1ABE, E58A.15 :9466	MOV LS[BIT3] TO WR[1]	
U 1ABF, B610.15 :9467	MOV LS[T9] TO WR[0]	
U 1AC0, 3640.95 :9468	JSR [CHECK.RESULT]	:CHECK FOR N SET
U 1AC1, 0869.3C :9469	JMP [LOOP.TF.5.92]	:LOOP ON ERROR IF ENABLED
	CLR LS[ERROR.MASK]	
	MOV LS[T8] TO WR[0]	:RETRIEVE Q
	MOV LS[#1] TO WR[1]	:EXPECTED = 1
	JSR [CHECK.RESULT]	:CHECK Q NOT WRITTEN

[illegible]

```

U 1AF3, 09AE.E4 :9528      JMP [LOOP.TF.5.97]          ;LOOP ON ERROR IF ENABLED
U 1AF4, 8A70.1C :9529      JSR [INC.OTHER]          ;INC 'OTHER' DATA TO 98
                                LOOP.TF.5.98:
U 1AF5, 367E.15 :9530      MOV LS[BIT31] TO WR[0]          ;WR0 = 80000000
U 1AF6, B69B.15 :9531      MOV LS[ALTER.EVEN] TO WR[2]      ;WR2 = 55555555
U 1AF7, 589E.15 :9532      MOV LS[ONES] TO Q              ; Q = -1
                                :
U 1AF8, A601.35 :9533      BIT WR[0] WITH WR[2],
U 1AF9, B67E.95 :9534      DT(LONG)&SET.ALU.CC
U 1AFA, 0869.3C :9535      MOV LS[BIT31] TO WR[1]
U 1AFB, 09AF.54 :9536      JSR [CHECK.RESULT]          ;CHECK WR0 NOT WRITTEN
U 1AFC, 2004.15 :9537      JMP [LOOP.TF.5.98]          ;LOOP ON ERROR IF ENABLED
U 1AFD, B69A.95 :9538      MOV WR[2] TO WR[0]
U 1AFE, 0869.3C :9539      MOV LS[ALTER.EVEN] TO WR[1]
U 1AFF, 09AF.54 :9540      JSR [CHECK.RESULT]          ;CHECK WR2 NOT WRITTEN
U 1B00, B7FC.15 :9541      JMP [LOOP.TF.5.98]          ;LOOP ON ERROR IF ENABLED
U 1B01, DF44.95 :9542      MOV LS[ALU.CC] TO WR[0]
U 1B02, BE8A.95 :9543      MCOM LS[BIT2] TO WR[1]
U 1B03, B644.95 :9544      MOV WR[1] TO LS[ERROR.MASK]
U 1B04, 0869.3C :9545      MOV LS[BIT2] TO WR[1]
U 1B05, 09AF.54 :9546      JSR [CHECK.RESULT]          ;EXPECTED = Z BIT SET
U 1B06, 8A70.1C :9547      JMP [LOOP.TF.5.98]          ;CHECK FOR Z BIT SET
U 1B07, E58A.15 :9548      JSR [INC.OTHER]          ;LOOP ON ERROR IF ENABLED
                                :INC 'OTHER' DATA TO 99
                                LOOP.TF.5.99:
U 1B08, 2F85.15 :9549      CLR LS[ERROR.MASK]
U 1B09, B640.15 :9550      CLR WR[2]
U 1B0A, 589E.15 :9551      MOV LS[#1] TO WR[0]          ;WR2 = 0
                                :WR0 = 1
                                :Q = -1
U 1B0B, 2644.35 :9552      MOV LS[ONES] TO Q
U 1B0C, 37FC.95 :9553      CMP WR[2] WITH WR[0],
U 1B0D, 3E12.95 :9554      DT(LONG)&SET.ALU.CC
U 1B0E, 3640.95 :9555      MOV LS[ALU.CC] TO WR[1]
U 1B0F, 0869.3C :9556      MOV WR[1] TO LS[T9]
U 1B10, 09B0.84 :9557      MOV LS[#1] TO WR[1]
U 1B11, 2004.15 :9558      JSR [CHECK.RESULT]          ;CHECK WR0 NOT WRITTEN
U 1B12, 2F82.95 :9559      JMP [LOOP.TF.5.99]          ;LOOP ON ERROR IF ENABLED
U 1B13, 0869.3C :9560      MOV WR[2] TO WR[0]
U 1B14, 09B0.84 :9561      CLR WR[1]
U 1B15, DF46.15 :9562      JSR [CHECK.RESULT]          ;CHECK WR2 NOT WRITTEN
U 1B16, 3E8A.15 :9563      JMP [LOOP.TF.5.99]          ;LOOP ON ERROR IF ENABLED
U 1B17, 3612.15 :9564      MCOM LS[BIT3] TO WR[0]
U 1B18, 3646.95 :9565      MOV WR[0] TO LS[ERROR.MASK]      ;MASK ALL BUT BIT #3 (TO LOOK AT N BIT)
U 1B19, 0869.3C :9566      MOV LS[T9] TO WR[0]
U 1B1A, 09B0.84 :9567      MOV LS[BIT3] TO WR[1]
U 1B1B, E58A.15 :9568      JSR [CHECK.RESULT]          ;EXPECTED = N BIT SET
U 1B1C, 8A70.1C :9569      JMP [LOOP.TF.5.99]          ;CHECK FOR N BIT SET
U 1B1D, 8A70.1C :9570      CLR LS[ERROR.MASK]          ;LOOP ON ERROR IF ENABLED
U 1B1E, 8A70.1C :9571      JSR [INC.OTHER]
U 1B1F, 8A70.1C :9572      JSR [INC.OTHER]          ;9A EMPTY
U 1B20, 8A70.1C :9573      JSR [INC.OTHER]          ;WR.PLUS.WR+1 PREVIOUSLY TESTED
U 1B21, 8A70.1C :9574      JSR [INC.OTHER]          ;9C EMPTY
U 1B22, 8A70.1C :9575      JSR [INC.OTHER]          ;9D EMPTY
                                :9E EMPTY
                                :9F EMPTY
                                :INC 'OTHER' DATA TO A0
                                LOOP.TF.5.A0:
U 1B23, B69E.15 :9576      MOV LS[ONES] TO WR[0]          ;WR0 = -1
U 1B24, 5840.15 :9577      MOV LS[#1] TO Q              ; Q = 1
  
```

22-ENKCB-1.0 : ENKCB.MIC
: ENKCB.MCR
: ENKCB.MIC

1 1
TEST F - DEST CTL R 3-MAR-82
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Rev 01.00 Page 208
TEST F - DEST CTL ROM And SRC.ALU CTL PAL Tests (M8390 CPU Module)

```
U 1B25, 2800,15 :9583      ADD WR[0] TO Q
U 1B26, A180,15 :9584      MOV Q TO WR[0]
U 1B27, 2F82,95 :9585      CLR WR[1]
U 1B28, 0869,3C :9586      JSR [CHECK.RESULT]
U 1B29, 09B2,34 :9587      JMP [LOOP.TF.5.A0]
U 1B2A, 8A70,1C :9588      JSR [INC.OTHER]
                        :EXPECTED = 0
                        :CHECK WR.PLUS.Q
                        :LOOP ON ERROR IF ENABLED
                        :INC 'OTHER' DATA TO A1
LOOP.TF.5.A1:
U 1B2B, 3642,15 :9590      MOV LS[#2] TO WR[0]
U 1B2C, 5840,15 :9591      MOV LS[#1] TO Q
U 1B2D, A840,15 :9592      SUB WR[0] FROM Q
U 1B2E, A180,15 :9593      MOV Q TO WR[0]
U 1B2F, 369E,95 :9594      MOV LS[ONES] TO WR[1]
U 1B30, 0869,3C :9595      JSR [CHECK.RESULT]
U 1B31, 89B2,B4 :9596      JMP [LOOP.TF.5.A1]
U 1B32, 8A70,1C :9597      JSR [INC.OTHER]
                        :EXPECTED = -1
                        :CHECK WR.FROM.Q
                        :LOOP ON ERROR IF ENABLED
                        :INC 'OTHER' DATA TO A2
LOOP.TF.5.A2:
U 1B33, B640,15 :9599      MOV LS[BIT0] TO WR[0]
U 1B34, D842,15 :9600      MOV LS[#2] TO Q
U 1B35, A880,15 :9601      SUB Q FROM WR[0] TO Q
U 1B36, A180,15 :9602      MOV Q TO WR[0]
U 1B37, 369E,95 :9603      MOV LS[ONES] TO WR[1]
U 1B38, 0869,3C :9604      JSR [CHECK.RESULT]
U 1B39, 89B3,34 :9605      JMP [LOOP.TF.5.A2]
U 1B3A, 8A70,1C :9606      JSR [INC.OTHER]
                        :EXPECTED = -1
                        :CHECK Q.FROM.WR.Q
                        :LOOP ON ERROR IF ENABLED
                        :INC 'OTHER' DATA TO A3
LOOP.TF.5.A3:
U 1B3B, B698,15 :9608      MOV LS[ALTER.ODD] TO WR[0]
U 1B3C, 2040,15 :9609      INC WR[0]
U 1B3D, D89A,15 :9610      MOV LS[ALTER.EVEN] TO Q
U 1B3E, 28C0,15 :9611      BIS WR[0] TO Q
U 1B3F, A180,15 :9612      MOV Q TO WR[0]
U 1B40, 369E,95 :9613      MOV LS[ONES] TO WR[1]
U 1B41, 0869,3C :9614      JSR [CHECK.RESULT]
U 1B42, 09B3,B4 :9615      JMP [LOOP.TF.5.A3]
U 1B43, 8A70,1C :9616      JSR [INC.OTHER]
                        :EXPECTED = FFFFFFFF
                        :CHECK WR.OR.Q
                        :LOOP ON ERROR IF ENABLED
                        :INC 'OTHER' DATA TO A4
LOOP.TF.5.A4:
U 1B44, 3698,95 :9618      MOV LS[ALTER.ODD] TO WR[1]
U 1B45, 589E,15 :9619      MOV LS[ONES] TO Q
U 1B46, 2902,15 :9620      AND WR[1] TO Q
U 1B47, A180,15 :9621      MOV Q TO WR[0]
U 1B48, 0869,3C :9622      JSR [CHECK.RESULT]
U 1B49, 89B4,44 :9623      JMP [LOOP.TF.5.A4]
U 1B4A, 8A70,1C :9624      JSR [INC.OTHER]
                        :CHECK WR.AND.Q
                        :LOOP ON ERROR IF ENABLED
                        :INC 'OTHER' DATA TO A5
LOOP.TF.5.A5:
U 1B4B, B698,15 :9626      MOV LS[ALTER.ODD] TO WR[0]
U 1B4C, 589E,15 :9627      MOV LS[ONES] TO Q
U 1B4D, 2940,15 :9628      BIC WR[0] TO Q
U 1B4E, A180,15 :9629      MOV Q TO WR[0]
U 1B4F, B69A,95 :9630      MOV LS[ALTER.EVEN] TO WR[1]
U 1B50, 0869,3C :9631      JSR [CHECK.RESULT]
U 1B51, 89B4,B4 :9632      JMP [LOOP.TF.5.A5]
U 1B52, 8A70,1C :9633      JSR [INC.OTHER]
                        :CHECK WR.MASK.Q
                        :LOOP ON ERROR IF ENABLED
                        :INC 'OTHER' DATA TO A6
LOOP.TF.5.A6:
U 1B53, 369A,15 :9635      MOV LS[ALTER.EVEN] TO WR[0]
U 1B54, 589E,15 :9636      MOV LS[ONES] TO Q
U 1B55, 2980,15 :9637      XOR WR[0] TO Q
                        :WR0 = 55555555
                        :Q = FFFFFFFF
```



```

U 1B56, A180,15 :9638      MOV Q TO WR[0]
U 1B57, 3698,95 :9639      MOV LS[ALTER.ODD] TO WR[1]
U 1B58, 0869,3C :9640      JSR [CHECK.RESULT]           ;CHECK WR.XOR.Q
U 1B59, 89B5,34 :9641      JMP [LOOP.TF.5.A6]           ;LOOP ON ERROR IF ENABLED
U 1B5A, 8A70,1C :9642      JSR [INC.OTHER]           ;A7 EMPTY
U 1B5B, 8A70,1C :9643      JSR [INC.OTHER]           ;INC "OTHER" DATA TO A8
                        :9644
LOOP.TF.5.A8:
U 1B5C, 369F,15 :9645      MOV LS[ONES] TO WR[2]           ;WR2 = -1
U 1B5D, B641,95 :9646      MOV LS[BIT0] TO WR[3]           ;WR3 = 1
U 1B5E, 5898,15 :9647      MOV LS[ALTER.ODD] TO Q
U 1B5F, 2A05,95 :9648      ADD WR[2] PLUS WR[3] TO Q
U 1B60, A180,15 :9649      MOV Q TO WR[0]
U 1B61, 2F82,95 :9650      CLR WR[1]
U 1B62, 0869,3C :9651      JSR [CHECK.RESULT]           ;EXPECTED = 0
U 1B63, 89B5,C4 :9652      JMP [LOOP.TF.5.A8]           ;CHECK WR.PLUS.WR.Q
U 1B64, 8A70,1C :9653      JSR [INC.OTHER]           ;LOOP ON ERROR IF ENABLED
                        :9654      JSR [INC.OTHER]           ;INC "OTHER" DATA TO A9
LOOP.TF.5.A9:
U 1B65, B643,15 :9655      MOV LS[#2] TO WR[2]           ;WR2 = 2
U 1B66, B641,95 :9656      MOV LS[BIT0] TO WR[3]           ;WR3 = 1
U 1B67, 5898,15 :9657      MOV LS[ALTER.ODD] TO Q
U 1B68, AA45,95 :9658      SUB WR[2] FROM WR[3] TO Q
U 1B69, A180,15 :9659      MOV Q TO WR[0]
U 1B6A, 369E,95 :9660      MOV LS[ONES] TO WR[1]           ;EXPECTED = -1
U 1B6B, 0869,3C :9661      JSR [CHECK.RESULT]           ;CHECK WR.FROM.WR.Q
U 1B6C, 89B6,54 :9662      JMP [LOOP.TF.5.A9]           ;LOOP ON ERROR IF ENABLED
U 1B6D, 8A70,1C :9663      JSR [INC.OTHER]           ;AA EMPTY
U 1B6E, 8A70,1C :9664      JSR [INC.OTHER]           ;INC "OTHER" DATA TO AB
                        :9665
LOOP.TF.5.AB:
U 1B6F, 3699,15 :9666      MOV LS[ALTER.ODD] TO WR[2]
U 1B70, 2045,15 :9667      INC WR[2]
                        :9668      ;WR2 = AAAAAAAB
U 1B71, 369B,95 :9669      MOV LS[ALTER.EVEN] TO WR[3]       ;WR3 = 55555555
U 1B72, D828,15 :9670      MOV LS[#FFFF] TO Q
U 1B73, 2AC5,95 :9671      BIS WR[2] WITH WR[3] TO Q
U 1B74, A180,15 :9672      MOV Q TO WR[0]
U 1B75, 369E,95 :9673      MOV LS[ONES] TO WR[1]           ;EXPECTED = FFFFFFFF
U 1B76, 0869,3C :9674      JSR [CHECK.RESULT]           ;CHECK WR.OR.WR.Q
U 1B77, 89B6,F4 :9675      JMP [LOOP.TF.5.AB]           ;LOOP ON ERROR IF ENABLED
U 1B78, 8A70,1C :9676      JSR [INC.OTHER]           ;INC "OTHER" DATA TO AC
                        :9677
LOOP.TF.5.AC:
U 1B79, 3699,15 :9678      MOV LS[ALTER.ODD] TO WR[2]
U 1B7A, 2045,15 :9679      INC WR[2]
                        :9680      ;WR2 = AAAAAAAB
U 1B7B, 369B,95 :9681      MOV LS[ALTER.EVEN] TO WR[3]       ;WR3 = 55555555
U 1B7C, D828,15 :9682      MOV LS[#FFFF] TO Q
U 1B7D, AB05,95 :9683      AND WR[2] WITH WR[3] TO Q
U 1B7E, A180,15 :9684      MOV Q TO WR[0]
U 1B7F, 3640,95 :9685      MOV LS[BIT0] TO WR[1]           ;EXPECTED = 00000001
U 1B80, 0869,3C :9686      JSR [CHECK.RESULT]           ;CHECK WR.AND.WR.Q
U 1B81, 09B7,94 :9687      JMP [LOOP.TF.5.AC]           ;LOOP ON ERROR IF ENABLED
U 1B82, 8A70,1C :9688      JSR [INC.OTHER]           ;INC "OTHER" DATA TO AD
                        :9689
LOOP.TF.5.AD:
U 1B83, 3699,15 :9690      MOV LS[ALTER.ODD] TO WR[2]
U 1B84, B69F,95 :9691      MOV LS[ONES] TO WR[3]
                        :9692      ;WR2 = AAAAAAAA
U 1B85, D828,15 :9693      MOV LS[#FFFF] TO Q
                        :9694      ;WR3 = FFFFFFFF
U 1B86, 2B45,95 :9695      BIC WR[2] WITH WR[3] TO Q
U 1B87, A180,15 :9696      MOV Q TO WR[0]
    
```

U 1B88.	B69A.95	:9693	MOV LS[ALTER.EVEN] TO WR[1]	:EXPECTED = 55555555
U 1B39.	0869.3C	:9694	JSR [CHECK.RESULT]	:CHECK WR.MASK.WR.Q
U 1B8A.	09B8.34	:9695	JMP [LOOP.TF.5.AE]	:LOOP ON ERROR IF ENABLED
U 1B8B.	8A70.1C	:9696	JSR [INC.OTHER]	:INC "OTHER" DATA TO AF
		:9697	LOOP.TF.5.AE:	
U 1B8C.	B69B.15	:9698	MOV LS[ALTER.EVEN] TO WR[2]	:WR2 = 55555555
U 1B8D.	B69F.95	:9699	MOV LS[ONES] TO WR[3]	:WR3 = FFFFFFFF
U 1B8E.	D828.15	:9700	MOV LS[FFFFFF] TO Q	
U 1B8F.	2B85.95	:9701	XOR WR[2] WITH WR[3] TO Q	
U 1B90.	A180.15	:9702	MOV Q TO WR[0]	
U 1B91.	3698.95	:9703	MOV LS[ALTER.ODD] TO WR[1]	:EXPECTED = AAAAAAAAAA
U 1B92.	0869.3C	:9704	JSR [CHECK.RESULT]	:CHECK WR.XOR.WR.Q
U 1B93.	09B8.C4	:9705	JMP [LOOP.TF.5.AE]	:LOOP ON ERROR IF ENABLED
U 1B94.	8A70.1C	:9706	JSR [INC.OTHER]	:AF EMPTY
U 1B95.	8A70.1C	:9707	JSR [INC.OTHER]	:INC "OTHER" DATA TO B0
		:9708	LOOP.TF.5.B0:	
U 1B96.	589E.15	:9709	MOV LS[ONES] TO Q	:Q = -1
U 1B97.	B640.15	:9710	MOV LS[B10] TO WR[0]	:WRO = 1
U 1B98.	AC00.15	:9711	ADD Q TO WR[0]	
U 1B99.	2F82.95	:9712	CLR WR[1]	:EXPECTED = 0
U 1B9A.	0869.3C	:9713	JSR [CHECK.RESULT]	:CHECK Q.PLUS.WR
U 1B9B.	89B9.64	:9714	JMP [LOOP.TF.5.B0]	:LOOP ON ERROR IF ENABLED
U 1B9C.	8A70.1C	:9715	JSR [INC.OTHER]	:INC "OTHER" DATA TO B1
		:9716	LOOP.TF.5.B1:	
U 1B9D.	B643.15	:9717	MOV LS[#2] TO WR[2]	:WR2 = 2
U 1B9E.	5840.15	:9718	MOV LS[B10] TO Q	:Q = 1
U 1B9F.	AC44.15	:9719	SUB WRA[2] FROM Q TO WRB[0]	
U 1BA0.	369E.95	:9720	MOV LS[ONES] TO WR[1]	:EXPECTED = -1
U 1BA1.	0869.3C	:9721	JSR [CHECK.RESULT]	:CHECK WR.FROM.Q.WR
U 1BA2.	09B9.D4	:9722	JMP [LOOP.TF.5.B1]	:LOOP ON ERROR IF ENABLED
U 1BA3.	8A70.1C	:9723	JSR [INC.OTHER]	:INC "OTHER" DATA TO B2
		:9724	LOOP.TF.5.B2:	
U 1BA4.	D842.15	:9725	MOV LS[#2] TO Q ;Q = 2	
U 1BA5.	B640.15	:9726	MOV LS[B10] TO WR[0]	:WRO = 1
U 1BA6.	2C80.15	:9727	SUB Q FROM WR[0]	
U 1BA7.	369E.95	:9728	MOV LS[ONES] TO WR[1]	:EXPECTED = -1
U 1BA8.	0869.3C	:9729	JSR [CHECK.RESULT]	:CHECK Q.FROM.WR
U 1BA9.	09BA.44	:9730	JMP [LOOP.TF.5.B2]	:LOOP ON ERROR IF ENABLED
U 1BAA.	8A70.1C	:9731	JSR [INC.OTHER]	:INC "OTHER" DATA TO B3
		:9732	LOOP.TF.5.B3:	
U 1BAB.	5898.15	:9733	MOV LS[ALTER.ODD] TO Q	
U 1BAC.	2540.15	:9734	INC Q	:Q = AAAAAAAB
U 1BAD.	369A.15	:9735	MOV LS[ALTER.EVEN] TO WR[0]	:WRO = 55555555
U 1BAE.	AC00.15	:9736	BIS Q TO WR[0]	
U 1BAF.	369E.95	:9737	MOV LS[ONES] TO WR[1]	:EXPECTED = FFFFFFFF
U 1BB0.	0869.3C	:9738	JSR [CHECK.RESULT]	:CHECK Q.OR.WR
U 1BB1.	09BA.B4	:9739	JMP [LOOP.TF.5.B3]	:LOOP ON ERROR IF ENABLED
U 1BB2.	8A70.1C	:9740	JSR [INC.OTHER]	:INC "OTHER" DATA TO B4
		:9741	LOOP.TF.5.B4:	
U 1BB3.	5898.15	:9742	MOV LS[ALTER.ODD] TO Q	
U 1BB4.	2540.15	:9743	INC Q	:Q = AAAAAAAB
U 1BB5.	369A.15	:9744	MOV LS[ALTER.EVEN] TO WR[0]	:WRO = 55555555
U 1BB6.	2000.15	:9745	AND Q TO WR[0]	
U 1BB7.	3640.95	:9746	MOV LS[B10] TO WR[1]	:EXPECTED = 00000001
U 1BB8.	0869.3C	:9747	JSR [CHECK.RESULT]	:CHECK Q.AND.WR

```

ZZ-ENKCB-1.0      : ENKCB.MIC      L 1
: ENKCB.MCR        MICRO2 1L(03)   3-MAR-82 10:02:46  ENKCB Micro-diagnostic for DAP module Rev 01.00  Page 211
: ENKCB.MIC        TEST F - DEST CTL ROM And SRC.ALU CTL PAL Tests (M8390 CPU Module)

U 1BB9, 09BB,34 :9748      JMP [LOOP.TF.5.B4]      ;LOOP ON ERROR IF ENABLED
U 1BBA, 8A70,1C :9749      JSR [INC.OTHER]      ;B5 EMPTY
U 1BBB, 8A70,1C :9750      JSR [INC.OTHER]      ;INC "OTHER" DATA TO B6
:9751      LOOP.TF.5.B6:
U 1BBC, 5898,15 :9752      MOV LS[ALTER.ODD] TO Q      ;Q = AAAAAAAA
U 1BBD, B69E,15 :9753      MOV LS[ONES] TO WR[0]      ;WRO = FFFFFFFF
U 1BBE, AD80,15 :9754      XOR Q TO WR[0]
U 1BBF, B69A,95 :9755      MOV LS[ALTER.EVEN] TO WR[1]      ;EXPECTED = 55555555
U 1BC0, 0869,3C :9756      JSR [CHECK.RESULT]      ;CHECK Q.XOR.WR
U 1BC1, 09BB,C4 :9757      JMP [LOOP.TF.5.B6]      ;LOOP ON ERROR IF ENABLED
U 1BC2, 8A70,1C :9758      JSR [INC.OTHER]      ;INC "OTHER" DATA TO B7
:9759      LOOP.TF.5.B7:
U 1BC3, D87E,15 :9760      MOV LS[BIT31] TO Q      ;Q = 800000C7
U 1BC4, 369A,15 :9761      MOV LS[ALTER.EVEN] TO WR[0]      ;WRO = 55555555
:9762      BIT Q WITH WR[0],
U 1BC5, ADC0,35 :9763      DT(LONG)&SET.ALU.CC
U 1BC6, 7610,15 :9764      MOV Q TO LS[T8]      ;SAVE Q
U 1BC7, B69A,95 :9765      MOV LS[ALTER.EVEN] TO WR[1]
U 1BC8, 0869,3C :9766      JSR [CHECK.RESULT]      ;CHECK WRO NOT WRITTEN
U 1BC9, 89BC,34 :9767      JMP [LOOP.TF.5.B7]      ;LOOP ON ERROR IF ENABLED
U 1BCA, B7FC,15 :9768      MOV LS[ALU.CC] TO WR[0]
U 1BCB, DF44,95 :9769      MCOM LS[BIT2] TO WR[1]
U 1BCC, BE8A,95 :9770      MOV WR[1] TO LS[ERROR.MASK]
U 1BCD, B644,95 :9771      MOV LS[BIT2] TO WR[1]      ;EXPECTED = Z SET
U 1BCE, 0869,3C :9772      JSR [CHECK.RESULT]      ;CHECK FOR Z SET
U 1BCF, 89BC,34 :9773      JMP [LOOP.TF.5.B7]      ;LOOP ON ERROR IF ENABLED
U 1BD0, E58A,15 :9774      CLR LS[ERROR.MASK]
U 1BD1, B610,15 :9775      MOV LS[T8] TO WR[0]      ;RETRIEVE Q
U 1BD2, B67E,95 :9776      MOV LS[BIT31] TO WR[1]
U 1BD3, 0869,3C :9777      JSR [CHECK.RESULT]      ;CHECK Q NOT WRITTEN
U 1BD4, 89BC,34 :9778      JMP [LOOP.TF.5.B7]      ;LOOP ON ERROR IF ENABLED
U 1BD5, 8A70,1C :9779      JSR [INC.OTHER]      ;B8 EMPTY
U 1BD6, 8A70,1C :9780      JSR [INC.OTHER]      ;B9 EMPTY
U 1BD7, 8A70,1C :9781      JSR [INC.OTHER]      ;INC "OTHER" DATA TO BA
:9782      LOOP.TF.5.BA:
U 1BD8, 3642,15 :9783      MOV LS[#2] TO WR[0]      ;WRO = 2
U 1BD9, 3641,15 :9784      MOV LS[BIT0] TO WR[2]      ;WR2 = 1
U 1BDA, D828,15 :9785      MOV LS[#FFFF] TO Q
U 1BDB, 2E84,15 :9786      SUB WRB[0] FROM WRA[2] TO WRB[0]
U 1BDC, 369E,95 :9787      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 1BDD, 0869,3C :9788      JSR [CHECK.RESULT]      ;CHECK WR.FROM.WR.WR
U 1BDE, 89BD,84 :9789      JMP [LOOP.TF.5.BA]      ;LOOP ON ERROR IF ENABLED
U 1BDF, 8A70,1C :9790      JSR [INC.OTHER]      ;INC "OTHER" DATA TO BB
:9791      LOOP.TF.5.BB:
U 1BE0, 3699,15 :9792      MOV LS[ALTER.ODD] TO WR[2]
U 1BE1, 2045,15 :9793      INC WR[2]      ;WR2 = AAA1AAAB
U 1BE2, 369A,15 :9794      MOV LS[ALTER.EVEN] TO WR[0]      ;WRO = 55555555
U 1BE3, D828,15 :9795      MOV LS[#FFFF] TO Q
U 1BE4, AEC4,15 :9796      BIS WR[2] TO WR[0]
U 1BE5, 369E,95 :9797      MOV LS[ONES] TO WR[1]      ;EXPECTED = FFFFFFFF
U 1BE6, 0869,3C :9798      JSR [CHECK.RESULT]      ;CHECK WR.OR.WR
U 1BE7, 09BE,04 :9799      JMP [LOOP.TF.5.BB]      ;LOOP ON ERROR IF ENABLED
U 1BE8, 8A70,1C :9800      JSR [INC.OTHER]      ;WR.FROM.WR-1 PREVIOUSLY TESTED
U 1BE9, 8A70,1C :9801      JSR [INC.OTHER]      ;INC "OTHER" DATA TO BD
:9802      LOOP.TF.5.BD:

```

```

U 1BEA, 3699,15 :9803      MOV LS[ALTER.ODD] TO WR[2]      ;WR2 = AAAAAAAA
U 1BEB, B69E,15 :9804      MOV LS[ONES] TO WR[0]        ;WRO = FFFFFFFF
U 1BEC, D828,15 :9805      MOV LS[FFFF] TO Q
U 1BED, AF44,15 :9806      BIC WR[2] TO WR[0]
U 1BEE, B69A,95 :9807      MOV LS[ALTER.EVEN] TO WR[1]    ;EXPECTED = 55555555
U 1BEF, 0869,3C :9808      JSR [CHECK.RESULT]           ;CHECK WR.MASK.WR
U 1BF0, 09BE,A4 :9809      JMP [LOOP.TF.5.BD]           ;LOOP ON ERROR IF ENABLED
U 1BF1, 8A70,1C :9810      JSR [INC.OTHER]              ;INC 'OTHER' DATA TO BE
:9811
LOOP.TF.5.BE:
U 1BF2, B69B,15 :9812      MOV LS[ALTER.EVEN] TO WR[2]    ;WR2 = 55555555
U 1BF3, B69E,15 :9813      MOV LS[ONES] TO WR[0]        ;WRO = FFFFFFFF
U 1BF4, D828,15 :9814      MOV LS[FFFF] TO Q
U 1BF5, AF84,15 :9815      XOR WR[2] TO WR[0]
U 1BF6, 3698,95 :9816      MOV LS[ALTER.ODD] TO WR[1]    ;EXPECTED = AAAAAAAA
U 1BF7, 0869,3C :9817      JSR [CHECK.RESULT]           ;CHECK WR.XOR.WR
U 1BF8, 09BF,24 :9818      JMP [LOOP.TF.5.BE]           ;LOOP ON ERROR IF ENABLED
U 1BF9, 8A70,1C :9819      JSR [INC.OTHER]              ;INC 'OTHER' DATA TO BF
:9820
LOOP.TF.5.BF:
U 1BFA, 3699,15 :9821      MOV LS[ALTER.ODD] TO WR[2]
U 1BFB, 2045,15 :9822      INC WR[2]                    ;WR2 = AAAAAAAB
U 1BFC, 369A,15 :9823      MOV LS[ALTER.EVEN] TO WR[0]    ;WRO = 55555555
U 1BFD, D828,15 :9824      MOV LS[FFFF] TO Q
U 1BFE, 2FC4,15 :9825      AND WR[2] TO WR[0]
U 1BFF, 3640,95 :9826      MOV LS[BIT0] TO WR[1]         ;EXPECTED = 00000001
U 1C00, 0869,3C :9827      JSR [CHECK.RESULT]           ;CHECK WR.AND.WR
U 1C01, 89BF,A4 :9828      JMP [LOOP.TF.5.BF]           ;LOOP ON ERROR IF ENABLED
U 1C02, 89C1,A4 :9829      JMP [TF.6]

```

: BASIC Instruction Tests

: The BASIC Instructions will be tested in groups of four, the same instruction
 : being tested for four different ranges of Local Store; 0-1F, 20-3F, 40-5F, and
 : 60-7F. The four actual locations used will be:

```

:9836      LS 8 - T8
:9837      LS 30 - #26
:9838      LS 53 - #F
:9839      and LS 73 - #3F

```

:9840
 :9841 1C1A:
 :9842 TF.6:

```

U 1C1A, 8870,0C :9843      JSR [INC.EN]                ;INCREMENT ERROR NUMBER TO 6
U 1C1B, 3660,15 :9844      MOV LS[L30] TO WR[0]          ;SAVE LS 30,53, AND 73
U 1C1C, B6A6,95 :9845      MOV LS[L53] TO WR[1]
U 1C1D, 36E7,15 :9846      MOV LS[L73] TO WR[2]
U 1C1E, BE12,15 :9847      MOV WR[0] TO LS[T9]
U 1C1F, 3E14,95 :9848      MOV WR[1] TO LS[T10]
U 1C20, BE17,15 :9849      MOV WR[2] TO LS[T11]
U 1C21, 3650,15 :9850      MOV LS[#100] TO WR[0]
U 1C22, BE88,15 :9851      MOV WR[0] TO LS[ADDRESS.DATA] ;INIT 'OTHER' DATA FIELD TO 100
U 1C23, 5F28,15 :9852      MCOM LS[FFFF] TO WR[0]
U 1C24, BE18,15 :9853      MOV WR[0] TO LS[T12]          ;T12 = FFFF0000 (MASK FOR UPPER WORD)
:9854
LOOP.TF.6.100:
U 1C25, AB80,15 :9855      CLR Q
U 1C26, 3642,15 :9856      MOV LS[#2] TO WR[0]
U 1C27, 3E10,15 :9857      MOV WR[0] TO LS[T8]          ;LS = 2

```

ZZ-ENKCB-1.0 : ENKCB.MIC
 : ENKCB.MCR
 : ENKCB.MIC

MICRO2 1L(03) TEST F - DEST CTL R 3-MAR-82
 TEST F - DEST CTL ROM And SRC.ALU CTL PAL Tests (M8390 CPU Module)

N 1
 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module
 Fiche 2 Frame N1
 Sequence 219
 Rev 01.00 Page 213

```

U 1C28, B69E,15 :9858      MOV LS[ONES] TO WR[0]          ;WRO = -1
U 1C29, 4010,15 :9859      ADD LS[T8] TO WR[0]
U 1C2A, 3640,95 :9860      MOV LS[#1] TO WR[1]          ;EXPECTED = 1
U 1C2B, 0869,3C :9861      JSR [CHECK.RESULT]          ;CHECK LS.PLUS.WR
U 1C2C, 89C2,54 :9862      JMP [LOOP.TF.6.100]          ;LOOP ON ERROR IF ENABLED
U 1C2D, 8A70,1C :9863      JSR [INC.OTHER]
                        :9864
LOOP.TF.6.101:
U 1C2E, AB80,15 :9865      CLR Q
U 1C2F, 3642,15 :9866      MOV LS[#2] TO WR[0]
U 1C30, BE60,15 :9867      MOV WR[0] TO LS[L30]          ;LS = 2
U 1C31, B69E,15 :9868      MOV LS[ONES] TO WR[0]        ;WRO = -1
U 1C32, C060,15 :9869      ADD LS[L30] TO WR[0]
U 1C33, 3640,95 :9870      MOV LS[#1] TO WR[1]          ;EXPECTED = 1
U 1C34, 0869,3C :9871      JSR [CHECK.RESULT]          ;CHECK LS.PLUS.WR
U 1C35, 09C2,E4 :9872      JMP [LOOP.TF.6.101]          ;LOOP ON ERROR IF ENABLED
U 1C36, 8A70,1C :9873      JSR [INC.OTHER]
                        :9874
LOOP.TF.6.102:
U 1C37, AB80,15 :9875      CLR Q
U 1C38, 3642,15 :9876      MOV LS[#2] TO WR[0]
U 1C39, BEA6,15 :9877      MOV WR[0] TO LS[L53]          ;LS = 2
U 1C3A, B69E,15 :9878      MOV LS[ONES] TO WR[0]        ;WRO = -1
U 1C3B, C0A6,15 :9879      ADD LS[L53] TO WR[0]
U 1C3C, 3640,95 :9880      MOV LS[#1] TO WR[1]          ;EXPECTED = 1
U 1C3D, 0869,3C :9881      JSR [CHECK.RESULT]          ;CHECK LS.PLUS.WR
U 1C3E, 89C3,74 :9882      JMP [LOOP.TF.6.102]          ;LOOP ON ERROR IF ENABLED
U 1C3F, 8A70,1C :9883      JSR [INC.OTHER]
                        :9884
LOOP.TF.6.103:
U 1C40, AB80,15 :9885      CLR Q
U 1C41, 3642,15 :9886      MOV LS[#2] TO WR[0]
U 1C42, 3EE6,15 :9887      MOV WR[0] TO LS[L73]          ;LS = 2
U 1C43, B69E,15 :9888      MOV LS[ONES] TO WR[0]        ;WRO = -1
U 1C44, 40E6,15 :9889      ADD LS[L73] TO WR[0]
U 1C45, 3640,95 :9890      MOV LS[#1] TO WR[1]          ;EXPECTED = 1
U 1C46, 0869,3C :9891      JSR [CHECK.RESULT]          ;CHECK LS.PLUS.WR
U 1C47, 89C4,04 :9892      JMP [LOOP.TF.6.103]          ;LOOP ON ERROR IF ENABLED
U 1C48, 8A70,1C :9893      JSR [INC.OTHER]
                        :9894
LOOP.TF.6.104:
U 1C49, AB80,15 :9895      CLR Q
U 1C4A, 3642,15 :9896      MOV LS[#2] TO WR[0]
U 1C4B, 3E10,15 :9897      MOV WR[0] TO LS[T8]
U 1C4C, B640,15 :9898      MOV LS[#1] TO WR[0]          ;LS = 2
U 1C4D, C110,15 :9899      SUB LS[T8] FROM WR[0]        ;WRO = 1
U 1C4E, 369E,95 :9900      MOV LS[ONES] TO WR[1]          ;EXPECTED = -1
U 1C4F, 0869,3C :9901      JSR [CHECK.RESULT]          ;CHECK LS.FROM.WR
U 1C50, 89C4,94 :9902      JMP [LOOP.TF.6.104]          ;LOOP ON ERROR IF ENABLED
U 1C51, 8A70,1C :9903      JSR [INC.OTHER]
                        :9904
LOOP.TF.6.105:
U 1C52, AB80,15 :9905      CLR Q
U 1C53, 3642,15 :9906      MOV LS[#2] TO WR[0]
U 1C54, BE60,15 :9907      MOV WR[0] TO LS[L30]          ;LS = 2
U 1C55, B640,15 :9908      MOV LS[#1] TO WR[0]          ;WRO = 1
U 1C56, 4160,15 :9909      SUB LS[L30] FROM WR[0]
U 1C57, 369E,95 :9910      MOV LS[ONES] TO WR[1]          ;EXPECTED = -1
U 1C58, 0869,3C :9911      JSR [CHECK.RESULT]          ;CHECK LS.FROM.WR
U 1C59, 89C5,24 :9912      JMP [LOOP.TF.6.105]          ;LOOP ON ERROR IF ENABLED
  
```

#####

[illegible]

[illegible]


```

:10078 LOOP.TF.6.116:
U 1CFE, AB80,15 :10079 CLR Q
U 1CF0, B698,15 :10080 MOV LS[ALTER.ODD] TO WR[0]
U 1CF1, BEA6,15 :10081 MOV WR[0] TO LS[L53] ; LS = AAAAAAAA
U 1CF2, B69E,15 :10082 MOV LS[ONES] TO WR[0] ; WRO = FFFFFFFF
U 1CF3, C5A6,15 :10083 BIC LS[L53] TO WR[0]
U 1CF4, B69A,95 :10084 MOV LS[ALTER.EVEN] TO WR[1] ; EXPECTED = 55555555
U 1CF5, 0869,3C :10085 JSR [CHECK.RESULT] ; CHECK LS.MASK.WR
U 1CF6, 89CE,F4 :10086 JMP [LOOP.TF.6.116] ; LOOP ON ERROR IF ENABLED
U 1CF7, 8A70,1C :10087 JSR [INC.OTHER]
:10088 LOOP.TF.6.117:
U 1CF8, AB80,15 :10089 CLR Q
U 1CF9, B698,15 :10090 MOV LS[ALTER.ODD] TO WR[0]
U 1CFA, 3EE6,15 :10091 MOV WR[0] TO LS[L73] ; LS = AAAAAAAA
U 1CFB, B69E,15 :10092 MOV LS[ONES] TO WR[0] ; WRO = FFFFFFFF
U 1CFC, 45E6,15 :10093 BIC LS[L73] TO WR[0]
U 1CFD, B69A,95 :10094 MOV LS[ALTER.EVEN] TO WR[1] ; EXPECTED = 55555555
U 1CFE, 0869,3C :10095 JSR [CHECK.RESULT] ; CHECK LS.MASK.WR
U 1CFF, 89CF,84 :10096 JMP [LOOP.TF.6.117] ; LOOP ON ERROR IF ENABLED
U 1D00, 8A70,1C :10097 JSR [INC.OTHER]
:10098 LOOP.TF.6.118:
U 1D01, AB80,15 :10099 CLR Q
U 1D02, B69E,15 :10100 MOV LS[ONES] TO WR[0]
U 1D03, 3E10,15 :10101 MOV WR[0] TO LS[L18] ; LS = -1
U 1D04, B640,15 :10102 MOV LS[#1] TO WR[0] ; WRO = 1
U 1D05, 4610,15 :10103 ADD (LS[L18] TO WR[0])+1
U 1D06, 3640,95 :10104 MOV LS[#1] TO WR[1] ; EXPECTED = 1
U 1D07, 0869,3C :10105 JSR [CHECK.RESULT] ; CHECK LS.PLUS.WR+1
U 1D08, 09D0,14 :10106 JMP [LOOP.TF.6.118] ; LOOP ON ERROR IF ENABLED
U 1D09, 8A70,1C :10107 JSR [INC.OTHER]
:10108 LOOP.TF.6.119:
U 1D0A, AB80,15 :10109 CLR Q
U 1D0B, B69E,15 :10110 MOV LS[ONES] TO WR[0]
U 1D0C, BE60,15 :10111 MOV WR[0] TO LS[L30] ; LS = -1
U 1D0D, B640,15 :10112 MOV LS[#1] TO WR[0] ; WRO = 1
U 1D0E, C660,15 :10113 ADD (LS[L30] TO WR[0])+1
U 1D0F, 3640,95 :10114 MOV LS[#1] TO WR[1] ; EXPECTED = 1
U 1D10, 0869,3C :10115 JSR [CHECK.RESULT] ; CHECK LS.PLUS.WR+1
U 1D11, 89D0,A4 :10116 JMP [LOOP.TF.6.119] ; LOOP ON ERROR IF ENABLED
U 1D12, 8A70,1C :10117 JSR [INC.OTHER]
:10118 LOOP.TF.6.11A:
U 1D13, AB80,15 :10119 CLR Q
U 1D14, B69E,15 :10120 MOV LS[ONES] TO WR[0]
U 1D15, BEA6,15 :10121 MOV WR[0] TO LS[L53] ; LS = -1
U 1D16, B640,15 :10122 MOV LS[#1] TO WR[0] ; WRO = 1
U 1D17, C6A6,15 :10123 ADD (LS[L53] TO WR[0])+1
U 1D18, 3640,95 :10124 MOV LS[#1] TO WR[1] ; EXPECTED = 1
U 1D19, 0869,3C :10125 JSR [CHECK.RESULT] ; CHECK LS.PLUS.WR+1
U 1D1A, 09D1,34 :10126 JMP [LOOP.TF.6.11A] ; LOOP ON ERROR IF ENABLED
U 1D1B, 8A70,1C :10127 JSR [INC.OTHER]
:10128 LOOP.TF.6.11B:
U 1D1C, AB80,15 :10129 CLR Q
U 1D1D, B69E,15 :10130 MOV LS[ONES] TO WR[0]
U 1D1E, 3EE6,15 :10131 MOV WR[0] TO LS[L73] ; LS = -1
U 1D1F, B640,15 :10132 MOV LS[#1] TO WR[0] ; WRO = 1
  
```

00000000000000000000000000000000

[illegible]

[illegible]

```

U 1DB6, 3642,15 :10298      MOV LS[#2] TO WR[0]          ;WRC = 2
U 1DB7, 3641,15 :10299      MOV LS[#1] TO WR[2]
U 1DB8, 3EA7,15 :10300      MOV WR[2] TO LS[L53]          ; LS = 1
U 1DB9, CAA6,15 :10301      SUB WR[0] FROM LS[L53] TO Q
U 1DBA, A180,15 :10302      MOV Q TO WR[0]                ;GET Q
U 1DBB, 369E,95 :10303      MOV LS[ONES] TO WR[1]          ;EXPECTED = -1
U 1DBC, 0869,3C :10304      JSR [CHECK.RESULT]             ;CHECK WR.FROM.LS.Q
U 1DBD, 09DB,54 :10305      JMP [LOOP.TF.6.12A]            ;LOOP ON ERROR IF ENABLED
U 1DBE, 8A70,1C :10306      JSR [INC.OTHER]
U 1DBF, AB80,15 :10307      LOOP.TF.6.12B:
U 1DC0, 3642,15 :10308      CLR Q
U 1DC1, 3641,15 :10309      MOV LS[#2] TO WR[0]          ;WRO = 2
U 1DC2, BEE7,15 :10310      MOV LS[#1] TO WR[2]
U 1DC3, 4AE6,15 :10311      MOV WR[2] TO LS[L73]          ; LS = 1
U 1DC4, A180,15 :10312      SUB WR[0] FROM LS[L73] TO Q
U 1DC5, 369E,95 :10313      MOV Q TO WR[0]                ;GET Q
U 1DC6, 0869,3C :10314      MOV LS[ONES] TO WR[1]          ;EXPECTED = -1
U 1DC7, 09DB,F4 :10315      JSR [CHECK.RESULT]             ;CHECK WR.FROM.LS.Q
U 1DC8, 8A70,1C :10316      JMP [LOOP.TF.6.12B]            ;LOOP ON ERROR IF ENABLED
U 1DC9, AB80,15 :10317      JSR [INC.OTHER]
U 1DCA, B698,15 :10318      LOOP.TF.6.12C:
U 1DCB, 2040,15 :10319      CLR Q
U 1DCC, B69B,15 :10320      MOV LS[ALTER.ODD] TO WR[0]
U 1DCD, BE11,15 :10321      INC WR[0]                    ;WRO = AAAAAAAB
U 1DCE, C810,15 :10322      MOV LS[ALTER.EVEN] TO WR[2]
U 1DCF, A180,15 :10323      MOV WR[2] TO LS[L18]          ; LS = 55555555
U 1DD0, 369E,95 :10324      BIS WR[0] WITH LS[L18] TO Q
U 1DD1, 0869,3C :10325      MOV Q TO WR[0]                ;GET Q
U 1DD2, 89DC,94 :10326      MOV LS[ONES] TO WR[1]          ;EXPECTED = FFFFFFFF
U 1DD3, 8A70,1C :10327      JSR [CHECK.RESULT]             ;CHECK WR.OR.LS.Q
U 1DD4, AB80,15 :10328      JMP [LOOP.TF.6.12C]            ;LOOP ON ERROR IF ENABLED
U 1DD5, B698,15 :10329      JSR [INC.OTHER]
U 1DD6, 2040,15 :10330      LOOP.TF.6.12D:
U 1DD7, B69B,15 :10331      CLR Q
U 1DD8, 3E61,15 :10332      MOV LS[ALTER.ODD] TO WR[0]
U 1DD9, 4B60,15 :10333      INC WR[0]                    ;WRO = AAAAAAAB
U 1DDA, A180,15 :10334      MOV LS[ALTER.EVEN] TO WR[2]
U 1DE0, B698,15 :10335      MOV WR[2] TO LS[L30]          ; LS = 55555555
U 1DE1, 2040,15 :10336      MOV WR[0] WITH LS[L30] TO Q
U 1DE2, B69B,15 :10337      MOV Q TO WR[0]                ;GET Q
U 1DE3, 3EA7,15 :10338      MOV LS[ONES] TO WR[1]          ;EXPECTED = FFFFFFFF
U 1DE4, 4BA6,15 :10339      JSR [CHECK.RESULT]             ;CHECK WR.OR.LS.Q
U 1DE5, A180,15 :10340      JMP [LOOP.TF.6.12D]            ;LOOP ON ERROR IF ENABLED
U 1DE6, 369E,95 :10341      JSR [INC.OTHER]
U 1DE7, 0869,3C :10342      LOOP.TF.6.12E:
U 1DE8, 09DD,F4 :10343      CLR Q
U 1DE9, B698,15 :10344      MOV LS[ALTER.ODD] TO WR[0]
U 1DEA, 2040,15 :10345      INC WR[0]                    ;WRO = AAAAAAAB
U 1DEB, B69B,15 :10346      MOV LS[ALTER.EVEN] TO WR[2]
U 1DEC, 3EA7,15 :10347      MOV WR[2] TO LS[L53]          ; LS = 55555555
U 1DED, 4BA6,15 :10348      BIS WR[0] WITH LS[L53] TO Q
U 1DEE, A180,15 :10349      MOV Q TO WR[0]                ;GET Q
U 1DEF, 369E,95 :10350      MOV LS[ONES] TO WR[1]          ;EXPECTED = FFFFFFFF
U 1DE8, 0869,3C :10351      JSR [CHECK.RESULT]             ;CHECK WR.OR.LS.Q
U 1DE9, 09DD,F4 :10352      JMP [LOOP.TF.6.12E]            ;LOOP ON ERROR IF ENABLED
    
```

```

U 1DE9, 8A70,1C :10353 JSR [INC.OTHER]
:10354 LOOP.TF.6.12F:
U 1DEA, AB80,15 :10355 CLR Q
U 1DEB, B698,15 :10356 MOV LS[ALTER.ODD] TO WR[0]
U 1DEC, 2040,15 :10357 INC WR[0] ;WRO = AAAAAAAB
U 1DED, B69B,15 :10358 MOV LS[ALTER.EVEN] TO WR[2] ;LS = 55555555
U 1DEE, BEE7,15 :10359 MOV WR[2] TO LS[L73]
U 1DEF, CBE6,15 :10360 BIS WR[0] WITH LS[L73] TO Q
U 1DF0, A180,15 :10361 MOV Q TO WR[0] ;GET Q
U 1DF1, 369E,95 :10362 MOV LS[ONES] TO WR[1] ;EXPECTED = FFFFFFFF
U 1DF2, 0869,3C :10363 JSR [CHECK.RESULT] ;CHECK WR.OR.LS.Q
U 1DF3, 09DE,A4 :10364 JMP [LOOP.TF.6.12F] ;LOOP ON ERROR IF ENABLED
U 1DF4, 8A70,1C :10365 JSR [INC.OTHER]
:10366 LOOP.TF.6.130:
U 1DF5, AB80,15 :10367 CLR Q
U 1DF6, B698,15 :10368 MOV LS[ALTER.ODD] TO WR[0] ;WRO = AAAAAAAA
U 1DF7, B629,15 :10369 MOV LS[FFFFFF] TO WR[2] ;LS = 0000FFFF
U 1DF8, BE11,15 :10370 MOV WR[2] TO LS[L8]
U 1DF9, 4C10,15 :10371 AND WR[0] WITH LS[L8] TO Q
U 1DFA, A180,15 :10372 MOV Q TO WR[0] ;GET Q
U 1DFB, 3698,95 :10373 MOV LS[ALTER.ODD] TO WR[1]
U 1DFC, 4518,95 :10374 BIC LS[L12] TO WR[1] ;EXPECTED = 0000AAAA
U 1DFD, 0869,3C :10375 JSR [CHECK.RESULT] ;CHECK WR.AND.LS.Q
U 1DFE, 89DF,54 :10376 JMP [LOOP.TF.6.130] ;LOOP ON ERROR IF ENABLED
U 1DFF, 8A70,1C :10377 JSR [INC.OTHER]
:10378 LOOP.TF.6.131:
U 1E00, AB80,15 :10379 CLR Q
U 1E01, B698,15 :10380 MOV LS[ALTER.ODD] TO WR[0] ;WRO = AAAAAAAA
U 1E02, B629,15 :10381 MOV LS[FFFFFF] TO WR[2] ;LS = 0000FFFF
U 1E03, 3E61,15 :10382 MOV WR[2] TO LS[L30]
U 1E04, CC60,15 :10383 AND WR[0] WITH LS[L30] TO Q
U 1E05, A180,15 :10384 MOV Q TO WR[0] ;GET Q
U 1E06, 3698,95 :10385 MOV LS[ALTER.ODD] TO WR[1]
U 1E07, 4518,95 :10386 BIC LS[L12] TO WR[1] ;EXPECTED = 0000AAAA
U 1E08, 0869,3C :10387 JSR [CHECK.RESULT] ;CHECK WR.AND.LS.Q
U 1E09, 89E0,04 :10388 JMP [LOOP.TF.6.131] ;LOOP ON ERROR IF ENABLED
U 1E0A, 8A70,1C :10389 JSR [INC.OTHER]
:10390 LOOP.TF.6.132:
U 1E0B, AB80,15 :10391 CLR Q
U 1E0C, B698,15 :10392 MOV LS[ALTER.ODD] TO WR[0] ;WRO = AAAAAAAA
U 1E0D, B629,15 :10393 MOV LS[FFFFFF] TO WR[2] ;LS = 0000FFFF
U 1E0E, 3EA7,15 :10394 MOV WR[2] TO LS[L53]
U 1E0F, CCA6,15 :10395 AND WR[0] WITH LS[L53] TO Q
U 1E10, A180,15 :10396 MOV Q TO WR[0] ;GET Q
U 1E11, 3698,95 :10397 MOV LS[ALTER.ODD] TO WR[1]
U 1E12, 4518,95 :10398 BIC LS[L12] TO WR[1] ;EXPECTED = 0000AAAA
U 1E13, 0869,3C :10399 JSR [CHECK.RESULT] ;CHECK WR.AND.LS.Q
U 1E14, 09E0,B4 :10400 JMP [LOOP.TF.6.132] ;LOOP ON ERROR IF ENABLED
U 1E15, 8A70,1C :10401 JSR [INC.OTHER]
:10402 LOOP.TF.6.133:
U 1E16, AB80,15 :10403 CLR Q
U 1E17, B698,15 :10404 MOV LS[ALTER.ODD] TO WR[0] ;WRO = AAAAAAAA
U 1E18, B629,15 :10405 MOV LS[FFFFFF] TO WR[2] ;LS = 0000FFFF
U 1E19, BEE7,15 :10406 MOV WR[2] TO LS[L73]
U 1E1A, 4CE6,15 :10407 AND WR[0] WITH LS[L73] TO Q
  
```

```

U 1E1B, A180,15 :10408      MOV Q TO WR[0]                ;GET Q
U 1E1C, 3698,95 :10409      MOV LS[ALTER.ODD] TO WR[1]
U 1E1D, 4518,95 :10410      BIC LS[T12] TO WR[1]          ;EXPECTED = 0000AAAA
U 1E1E, 0869,3C :10411      JSR [CHECK.RESULT]           ;CHECK WR.AND.LS.Q
U 1E1F, 09E1,64 :10412      JMP [LOOP.TF.6.133]          ;LOOP ON ERROR IF ENABLED
U 1E20, 8A70,1C :10413      JSR [INC.OTHER]
                        :10414
LOOP.TF.6.134:
U 1E21, AB80,15 :10415      CLR Q
U 1E22, B698,15 :10416      MOV LS[ALTER.ODD] TO WR[0]      ;WRO = AAAAAAAA
U 1E23, B629,15 :10417      MOV LS[FFFF] TO WR[2]
U 1E24, BE11,15 :10418      MOV WR[2] TO LS[T8]           ;LS = 0000FFFF
U 1E25, CD10,15 :10419      XOR WR[0] WITH LS[T8] TO Q
U 1E26, A180,15 :10420      MOV Q TO WR[0]
U 1E27, 3722,95 :10421      MOV LS[ALTER.MIX] TO WR[1]      ;GET Q
U 1E28, 0869,3C :10422      JSR [CHECK.RESULT]           ;EXPECTED = AAAA5555
U 1E29, 89E2,14 :10423      JMP [LOOP.TF.6.134]          ;CHECK WR.XOR.LS.Q
U 1E2A, 8A70,1C :10424      JSR [INC.OTHER]              ;LOOP ON ERROR IF ENABLED
                        :10425
LOOP.TF.6.135:
U 1E2B, AB80,15 :10426      CLR Q
U 1E2C, B698,15 :10427      MOV LS[ALTER.ODD] TO WR[0]      ;WRO = AAAAAAAA
U 1E2D, B629,15 :10428      MOV LS[FFFF] TO WR[2]
U 1E2E, 3E61,15 :10429      MOV WR[2] TO LS[L30]           ;LS = 0000FFFF
U 1E2F, 4D60,15 :10430      XOR WR[0] WITH LS[L30] TO Q
U 1E30, A180,15 :10431      MOV Q TO WR[0]
U 1E31, 3722,95 :10432      MOV LS[ALTER.MIX] TO WR[1]      ;GET Q
U 1E32, 0869,3C :10433      JSR [CHECK.RESULT]           ;EXPECTED = AAAA5555
U 1E33, 89E2,B4 :10434      JMP [LOOP.TF.6.135]          ;CHECK WR.XOR.LS.Q
U 1E34, 8A70,1C :10435      JSR [INC.OTHER]              ;LOOP ON ERROR IF ENABLED
                        :10436
LOOP.TF.6.136:
U 1E35, AB80,15 :10437      CLR Q
U 1E36, B698,15 :10438      MOV LS[ALTER.ODD] TO WR[0]      ;WRO = AAAAAAAA
U 1E37, B629,15 :10439      MOV LS[FFFF] TO WR[2]
U 1E38, 3EA7,15 :10440      MOV WR[2] TO LS[L53]           ;LS = 0000FFFF
U 1E39, 4DA6,15 :10441      XOR WR[0] WITH LS[L53] TO Q
U 1E3A, A180,15 :10442      MOV Q TO WR[0]
U 1E3B, 3722,95 :10443      MOV LS[ALTER.MIX] TO WR[1]      ;GET Q
U 1E3C, 0869,3C :10444      JSR [CHECK.RESULT]           ;EXPECTED = AAAA5555
U 1E3D, 89E3,54 :10445      JMP [LOOP.TF.6.136]          ;CHECK WR.XOR.LS.Q
U 1E3E, 8A70,1C :10446      JSR [INC.OTHER]              ;LOOP ON ERROR IF ENABLED
                        :10447
LOOP.TF.6.137:
U 1E3F, AB80,15 :10448      CLR Q
U 1E40, B698,15 :10449      MOV LS[ALTER.ODD] TO WR[0]      ;WRO = AAAAAAAA
U 1E41, B629,15 :10450      MOV LS[FFFF] TO WR[2]
U 1E42, BEE7,15 :10451      MOV WR[2] TO LS[L73]           ;LS = 0000FFFF
U 1E43, CDE6,15 :10452      XOR WR[0] WITH LS[L73] TO Q
U 1E44, A180,15 :10453      MOV Q TO WR[0]
U 1E45, 3722,95 :10454      MOV LS[ALTER.MIX] TO WR[1]      ;GET Q
U 1E46, 0869,3C :10455      JSR [CHECK.RESULT]           ;EXPECTED = AAAA5555
U 1E47, 89E3,F4 :10456      JMP [LOOP.TF.6.137]          ;CHECK WR.XOR.LS.Q
U 1E48, 8A70,1C :10457      JSR [INC.OTHER]              ;LOOP ON ERROR IF ENABLED
U 1E49, DF46,15 :10458      MCOM LS[BIT3] TO WR[0]
U 1E4A, 3E8A,15 :10459      MOV WR[0] TO LS[ERROR.MASK]      ;MASK ALL BUT BIT #3 (TO LOOK AT N BIT)
                        :10460
LOOP.TF.6.138:
U 1E4B, AB80,15 :10461      CLR Q
U 1E4C, B69E,15 :10462      MOV LS[ONES] TO WR[0]
  
```

```

U 1E4D, 3E10,15 :10463      MOV WR[0] TO LS[T8]                ; LS = -1
U 1E4E, B640,15 :10464      MOV LS[#1] TO WR[0]            ; WRO = 1
                                CMP LS[T8] WITH WR[0],
U 1E4F, 4E10,35 :10465      DT(LONG)&SET,ALU.CC
U 1E50, B7FC,15 :10467      MOV LS[ALU.CC] TO WR[0]
U 1E51, 3646,95 :10468      MOV LS[BIT3] TO WR[1]                ; EXPECTED = N SET
U 1E52, 0869,3C :10469      JSR [CHECK.RESULT]                ; CHECK LS.CMP.WR
U 1E53, 89E4,B4 :10470      JMP [LOOP.TF.6.138]                ; LOOP ON ERROR IF ENABLED
U 1E54, 8A70,1C :10471      JSR [INC.OTHER]
                                :10472
U 1E55, AB80,15 :10473      LOOP.TF.6.139:
                                CLR Q
U 1E56, B69E,15 :10474      MOV LS[ONES] TO WR[0]
U 1E57, BE60,15 :10475      MOV WR[0] TO LS[L30]                ; LS = -1
U 1E58, B640,15 :10476      MOV LS[#1] TO WR[0]                ; WRO = 1
                                :10477
                                CMP LS[L30] WITH WR[0],
U 1E59, CE60,35 :10478      DT(LONG)&SET,ALU.CC
U 1E5A, B7FC,15 :10479      MOV LS[ALU.CC] TO WR[0]
U 1E5B, 3646,95 :10480      MOV LS[BIT3] TO WR[1]                ; EXPECTED = N SET
U 1E5C, 0869,3C :10481      JSR [CHECK.RESULT]                ; CHECK LS.CMP.WR
U 1E5D, 89E5,54 :10482      JMP [LOOP.TF.6.139]                ; LOOP ON ERROR IF ENABLED
U 1E5E, 8A70,1C :10483      JSR [INC.OTHER]
                                :10484
U 1E5F, AB80,15 :10485      LOOP.TF.6.13A:
                                CLR Q
U 1E60, B69E,15 :10486      MOV LS[ONES] TO WR[0]
U 1E61, BEA6,15 :10487      MOV WR[0] TO LS[L53]                ; LS = -1
U 1E62, B640,15 :10488      MOV LS[#1] TO WR[0]                ; WRO = 1
                                :10489
                                CMP LS[L53] WITH WR[0],
U 1E63, CEA6,35 :10490      DT(LONG)&SET,ALU.CC
U 1E64, B7FC,15 :10491      MOV LS[ALU.CC] TO WR[0]
U 1E65, 3646,95 :10492      MOV LS[BIT3] TO WR[1]                ; EXPECTED = N SET
U 1E66, 0869,3C :10493      JSR [CHECK.RESULT]                ; CHECK LS.CMP.WR
U 1E67, 89E5,F4 :10494      JMP [LOOP.TF.6.13A]                ; LOOP ON ERROR IF ENABLED
U 1E68, 8A70,1C :10495      JSR [INC.OTHER]
                                :10496
U 1E69, AB80,15 :10497      LOOP.TF.6.13B:
                                CLR Q
U 1E6A, B69E,15 :10498      MOV LS[ONES] TO WR[0]
U 1E6B, 3EE6,15 :10499      MOV WR[0] TO LS[L73]                ; LS = -1
U 1E6C, B640,15 :10500      MOV LS[#1] TO WR[0]                ; WRO = 1
                                :10501
                                CMP LS[L73] WITH WR[0],
U 1E6D, 4EE6,35 :10502      DT(LONG)&SET,ALU.CC
U 1E6E, B7FC,15 :10503      MOV LS[ALU.CC] TO WR[0]
U 1E6F, 3646,95 :10504      MOV LS[BIT3] TO WR[1]                ; EXPECTED = N SET
U 1E70, 0869,3C :10505      JSR [CHECK.RESULT]                ; CHECK LS.CMP.WR
U 1E71, 89E6,94 :10506      JMP [LOOP.TF.6.13B]                ; LOOP ON ERROR IF ENABLED
U 1E72, 8A70,1C :10507      JSR [INC.OTHER]
                                :10508
U 1E73, AB80,15 :10509      LOOP.TF.6.13C:
                                CLR Q
U 1E74, B69E,15 :10510      MOV LS[ONES] TO WR[0]                ; WRO = -1
U 1E75, 3641,15 :10511      MOV LS[#1] TO WR[2]
U 1E76, BE11,15 :10512      MOV WR[2] TO LS[T8]                ; LS = 1
                                :10513
                                CMP WR[0] WITH LS[T8],
U 1E77, CF10,35 :10514      DT(LONG)&SET,ALU.CC
U 1E78, B7FC,15 :10515      MOV LS[ALU.CC] TO WR[0]
U 1E79, 3646,95 :10516      MOV LS[BIT3] TO WR[1]
U 1E7A, 0869,3C :10517      JSR [CHECK.RESULT]                ; CHECK WR.CMP.LS

```



```

U 1E7B, 09E7,34 :10518      JMP [LOOP.TF.6.13C]          ;LOOP ON ERROR IF ENABLED
U 1E7C, 8A70,1C :10519      JSR [INC.OTHER]
                                :10520
U 1E7D, AB80,15 :10521      LOOP.TF.6.13D:
U 1E7E, B69E,15 :10522      CLR Q
U 1E7F, 3641,15 :10523      MOV LS[ONES] TO WR[0]          ;WR0 = -1
U 1E80, 3E61,15 :10524      MOV LS[#1] TO WR[2]
                                :10525      MOV WR[2] TO LS[L30]          ; LS = 1
                                :10526      CMP WR[0] WITH LS[L30],
                                :10527      DT(LONG)&SET,ALU.CC
U 1E81, 4F60,35 :10528      MOV LS[ALU.CC] TO WR[0]
U 1E82, B7FC,15 :10529      MOV LS[BIT3] TO WR[1]
U 1E83, 3646,95 :10530      JSR [CHECK.RESULT]
U 1E84, 0869,3C :10531      JMP [LOOP.TF.6.13D]          ;CHECK WR.CMP.LS
U 1E85, 89E7,D4 :10532      JSR [INC.OTHER]          ;LOOP ON ERROR IF ENABLED
U 1E86, 8A70,1C :10533
                                :10534
U 1E87, AB80,15 :10535      LOOP.TF.6.13E:
U 1E88, B69E,15 :10536      CLR Q
U 1E89, 3641,15 :10537      MOV LS[ONES] TO WR[0]          ;WR0 = -1
U 1E8A, 3EA7,15 :10538      MOV LS[#1] TO WR[2]
                                :10539      MOV WR[2] TO LS[L53]          ; LS = 1
                                :10540      CMP WR[0] WITH LS[L53],
                                :10541      DT(LONG)&SET,ALU.CC
U 1E8B, 4FA6,35 :10542      MOV LS[ALU.CC] TO WR[0]
U 1E8C, B7FC,15 :10543      MOV LS[BIT3] TO WR[1]
U 1E8D, 3646,95 :10544      JSR [CHECK.RESULT]
U 1E8E, 0869,3C :10545      JMP [LOOP.TF.6.13E]          ;CHECK WR.CMP.LS
U 1E8F, 89E8,74 :10546      JSR [INC.OTHER]          ;LOOP ON ERROR IF ENABLED
U 1E90, 8A70,1C :10547
                                :10548
U 1E91, AB80,15 :10549      LOOP.TF.6.13F:
U 1E92, B69E,15 :10550      CLR Q
U 1E93, 3641,15 :10551      MOV LS[ONES] TO WR[0]          ;WR0 = -1
U 1E94, BEE7,15 :10552      MOV LS[#1] TO WR[2]
                                :10553      MOV WR[2] TO LS[L73]          ; LS = 1
                                :10554      CMP WR[0] WITH LS[L73],
                                :10555      DT(LONG)&SET,ALU.CC
U 1E95, CFE6,35 :10556      MOV LS[ALU.CC] TO WR[0]
U 1E96, B7FC,15 :10557      MOV LS[BIT3] TO WR[1]
U 1E97, 3646,95 :10558      JSR [CHECK.RESULT]
U 1E98, 0869,3C :10559      JMP [LOOP.TF.6.13F]          ;CHECK WR.CMP.LS
U 1E99, 09E9,14 :10560      JSR [INC.OTHER]          ;LOOP ON ERROR IF ENABLED
U 1E9A, 8A70,1C :10561      CLR LS[ERROR.MASK]
U 1E9B, E58A,15 :10562
                                :10563
U 1E9C, 3642,15 :10564      LOOP.TF.6.140:
U 1E9D, 3E10,15 :10565      MOV LS[#2] TO WR[0]
U 1E9E, 589E,15 :10566      MOV WR[0] TO LS[L8]
                                :10567      MOV LS[ONES] TO Q
                                :10568      ADD LS[L8] TO Q
                                :10569      MOV Q TO WR[0]
                                :10570      MOV LS[#1] TO WR[1]
                                :10571      JSR [CHECK.RESULT]
                                :10572      JMP [LOOP.TF.6.140]          ;EXPECTED = 1
                                :10573      JSR [INC.OTHER]          ;CHECK LS.PLUS.Q
                                :10574
                                :10575      LOOP.TF.6.141:
                                :10576      MOV LS[#2] TO WR[0]
                                :10577      MOV WR[0] TO LS[L30]
                                :10578      MOV LS[ONES] TO Q
                                :10579      ADD LS[L30] TO Q
                                :10580      MOV Q TO WR[0]
                                :10581
                                :10582
                                :10583
                                :10584
                                :10585
                                :10586
                                :10587
                                :10588
                                :10589
                                :10590
                                :10591
                                :10592
                                :10593
                                :10594
                                :10595
                                :10596
                                :10597
                                :10598
                                :10599
                                :10600
  
```

```

U 1EAA, 3640,95 :10573      MOV LS[#1] TO WR[1]      ;EXPECTED = 1
U 1EAB, 0869,3C :10574      JSR [CHECK.RESULT]      ;CHECK LS.PLUS.Q
U 1EAC, 89EA,54 :10575      JMP [LOOP.TF.6.141]      ;LOOP ON ERROR IF ENABLED
U 1EAD, 8A70,1C :10576      JSR [INC.OTHER]
                                LOOP.TF.6.142:
U 1EAE, 3642,15 :10578      MOV LS[#2] TO WR[0]
U 1EAF, BEA6,15 :10579      MOV WR[0] TO LS[L53]      ;LS = 2
U 1EB0, 589E,15 :10580      MOV LS[ONES] TO Q      ;Q = -1
U 1EB1, 50A6,15 :10581      ADD LS[L53] TO Q
U 1EB2, A180,15 :10582      MOV Q TO WR[0]
U 1EB3, 3640,95 :10583      MOV LS[#1] TO WR[1]      ;EXPECTED = 1
U 1EB4, 0869,3C :10584      JSR [CHECK.RESULT]      ;CHECK LS.PLUS.Q
U 1EB5, 09EA,E4 :10585      JMP [LOOP.TF.6.142]      ;LOOP ON ERROR IF ENABLED
U 1EB6, 8A70,1C :10586      JSR [INC.OTHER]
                                LOOP.TF.6.143:
U 1EB7, 3642,15 :10588      MOV LS[#2] TO WR[0]
U 1EB8, 3EE6,15 :10589      MOV WR[0] TO LS[L73]      ;LS = 2
U 1EB9, 589E,15 :10590      MOV LS[ONES] TO Q      ;Q = -1
U 1EBA, D0E6,15 :10591      ADD LS[L73] TO Q
U 1EBB, A180,15 :10592      MOV Q TO WR[0]
U 1EBC, 3640,95 :10593      MOV LS[#1] TO WR[1]      ;EXPECTED = 1
U 1EBD, 0869,3C :10594      JSR [CHECK.RESULT]      ;CHECK LS.PLUS.Q
U 1EBE, 89EB,74 :10595      JMP [LOOP.TF.6.143]      ;LOOP ON ERROR IF ENABLED
U 1EBF, 8A70,1C :10596      JSR [INC.OTHER]
                                LOOP.TF.6.144:
U 1EC0, 3642,15 :10598      MOV LS[#2] TO WR[0]
U 1EC1, 3E10,15 :10599      MOV WR[0] TO LS[T8]      ;LS = 2
U 1EC2, 5840,15 :10600      MOV LS[#1] TO Q      ;Q = 1
U 1EC3, 5110,15 :10601      SUB LS[T8] FROM Q
U 1EC4, A180,15 :10602      MOV Q TO WR[0]
U 1EC5, 369E,95 :10603      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 1EC6, 0869,3C :10604      JSR [CHECK.RESULT]      ;CHECK LS.FROM.Q
U 1EC7, 89EC,04 :10605      JMP [LOOP.TF.6.144]      ;LOOP ON ERROR IF ENABLED
U 1EC8, 8A70,1C :10606      JSR [INC.OTHER]
                                LOOP.TF.6.145:
U 1EC9, 3642,15 :10608      MOV LS[#2] TO WR[0]
U 1ECA, BE60,15 :10609      MOV WR[0] TO LS[L30]      ;LS = 2
U 1ECB, 5840,15 :10610      MOV LS[#1] TO Q      ;Q = 1
U 1ECC, D160,15 :10611      SUB LS[L30] FROM Q
U 1ECD, A180,15 :10612      MOV Q TO WR[0]
U 1ECE, 369E,95 :10613      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 1ECF, 0869,3C :10614      JSR [CHECK.RESULT]      ;CHECK LS.FROM.Q
U 1ED0, 89EC,94 :10615      JMP [LOOP.TF.6.145]      ;LOOP ON ERROR IF ENABLED
U 1ED1, 8A70,1C :10616      JSR [INC.OTHER]
                                LOOP.TF.6.146:
U 1ED2, 3642,15 :10618      MOV LS[#2] TO WR[0]
U 1ED3, BEA6,15 :10619      MOV WR[0] TO LS[L53]      ;LS = 2
U 1ED4, 5840,15 :10620      MOV LS[#1] TO Q      ;Q = 1
U 1ED5, D1A6,15 :10621      SUB LS[L53] FROM Q
U 1ED6, A180,15 :10622      MOV Q TO WR[0]
U 1ED7, 369E,95 :10623      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 1ED8, 0869,3C :10624      JSR [CHECK.RESULT]      ;CHECK LS.FROM.Q
U 1ED9, 89ED,24 :10625      JMP [LOOP.TF.6.146]      ;LOOP ON ERROR IF ENABLED
U 1EDA, 8A70,1C :10626      JSR [INC.OTHER]
                                LOOP.TF.6.147:
  
```

```

U 1EDB. 3642.15 :10628      MOV LS[#2] TO WR[0]
U 1EDC. 3EE6.15 :10629      MOV WR[0] TO LS[L73]      ;LS = 2
U 1EDD. 5840.15 :10630      MOV LS[#1] TO Q      ;Q = 1
U 1EDE. 51E6.15 :10631      SUB LS[L73] FROM Q
U 1EDF. A180.15 :10632      MOV Q TO WR[0]
U 1EE0. 369E.95 :10633      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 1EE1. 0869.3C :10634      JSR [CHECK.RESULT]      ;CHECK LS.FROM.Q
U 1EE2. 89ED.B4 :10635      JMP [LOOP.TF.6.147]      ;LOOP ON ERROR IF ENABLED
U 1EE3. 8A70.1C :10636      JSR [INC.OTHER]
      :10637
      LOOP.TF.6.148:
U 1EE4. D842.15 :10638      MOV LS[#2] TO Q      ;Q = 2
U 1EE5. B640.15 :10639      MOV LS[#1] TO WR[0]
U 1EE6. 3E10.15 :10640      MOV WR[0] TO LS[L73]      ;LS = 1
U 1EE7. 5210.15 :10641      SUB Q FROM LS[L73] TO Q
U 1EE8. A180.15 :10642      MOV Q TO WR[0]
U 1EE9. 369E.95 :10643      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 1EEA. 0869.3C :10644      JSR [CHECK.RESULT]      ;CHECK Q.FROM.LS.Q
U 1EEB. 89EE.44 :10645      JMP [LOOP.TF.6.148]      ;LOOP ON ERROR IF ENABLED
U 1EEC. 8A70.1C :10646      JSR [INC.OTHER]
      :10647
      LOOP.TF.6.149:
U 1EED. D842.15 :10648      MOV LS[#2] TO Q      ;Q = 2
U 1EEE. B640.15 :10649      MOV LS[#1] TO WR[0]
U 1EEF. BE60.15 :10650      MOV WR[0] TO LS[L30]      ;LS = 1
U 1EF0. D260.15 :10651      SUB Q FROM LS[L30] TO Q
U 1EF1. A180.15 :10652      MOV Q TO WR[0]
U 1EF2. 369E.95 :10653      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 1EF3. 0869.3C :10654      JSR [CHECK.RESULT]      ;CHECK Q.FROM.LS.Q
U 1EF4. 89EE.D4 :10655      JMP [LOOP.TF.6.149]      ;LOOP ON ERROR IF ENABLED
U 1EF5. 8A70.1C :10656      JSR [INC.OTHER]
      :10657
      LOOP.TF.6.14A:
U 1EF6. D842.15 :10658      MOV LS[#2] TO Q      ;Q = 2
U 1EF7. B640.15 :10659      MOV LS[#1] TO WR[0]
U 1EF8. BEA6.15 :10660      MOV WR[0] TO LS[L53]      ;LS = 1
U 1EF9. D2A6.15 :10661      SUB Q FROM LS[L53] TO Q
U 1EFA. A180.15 :10662      MOV Q TO WR[0]
U 1EFB. 369E.95 :10663      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 1EFC. 0869.3C :10664      JSR [CHECK.RESULT]      ;CHECK Q.FROM.LS.Q
U 1EFD. 89EF.64 :10665      JMP [LOOP.TF.6.14A]      ;LOOP ON ERROR IF ENABLED
U 1EFE. 8A70.1C :10666      JSR [INC.OTHER]
      :10667
      LOOP.TF.6.14B:
U 1EFF. D842.15 :10668      MOV LS[#2] TO Q      ;Q = 2
U 1F00. B640.15 :10669      MOV LS[#1] TO WR[0]
U 1F01. 3EE6.15 :10670      MOV WR[0] TO LS[L73]      ;LS = 1
U 1F02. 52E6.15 :10671      SUB Q FROM LS[L73] TO Q
U 1F03. A180.15 :10672      MOV Q TO WR[0]
U 1F04. 369E.95 :10673      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 1F05. 0869.3C :10674      JSR [CHECK.RESULT]      ;CHECK Q.FROM.LS.Q
U 1F06. 89EF.F4 :10675      JMP [LOOP.TF.6.14B]      ;LOOP ON ERROR IF ENABLED
U 1F07. 8A70.1C :10676      JSR [INC.OTHER]
      :10677
      LOOP.TF.6.14C:
U 1F08. B698.15 :10678      MOV LS[ALTER.ODD] TO WR[0]
U 1F09. 2040.15 :10679      INC WR[0]
U 1FOA. 3E10.15 :10680      MOV WR[0] TO LS[L73]      ;LS = AAAAAAAB
U 1FOB. D89A.15 :10681      MOV LS[ALTER.EVEN] TO Q      ;Q = 55555555
U 1F0C. D310.15 :10682      BIS LS[L73] TO Q
  
```

```

U 1F0D, A180,15 :10683      MOV Q TO WR[0]
U 1F0E, 369E,95 :10684      MOV LS[ONES] TO WR[1]      ;EXPECTED = FFFFFFFF
U 1F0F, 0869,3C :10685      JSR [CHECK.RESULT]      ;CHECK LS.OR.Q
U 1F10, 89F0,84 :10686      JMP [LOOP.TF.6.14C]      ;LOOP ON ERROR IF ENABLED
U 1F11, 8A70,1C :10687      JSR [INC.OTHER]
U 1F12, B698,15 :10688      LOOP.TF.6.14D:
U 1F13, 2040,15 :10689      MOV LS[ALTER.ODD] TO WR[0]
U 1F14, BE60,15 :10690      INC WR[0]
U 1F15, D89A,15 :10691      MOV WR[0] TO LS[L30]      ;LS = AAAAAAAB
U 1F16, 5360,15 :10692      MOV LS[ALTER.EVEN] TO Q      ;Q = 55555555
U 1F17, A180,15 :10693      BIS LS[L30] TO Q
U 1F18, 369E,95 :10694      MOV Q TO WR[0]
U 1F19, 0869,3C :10695      MOV LS[ONES] TO WR[1]      ;EXPECTED = FFFFFFFF
U 1F1A, 09F1,24 :10696      JSR [CHECK.RESULT]      ;CHECK LS.OR.Q
U 1F1B, 8A70,1C :10697      JMP [LOOP.TF.6.14D]      ;LOOP ON ERROR IF ENABLED
U 1F1C, B698,15 :10698      JSR [INC.OTHER]
U 1F1D, 2040,15 :10699      LOOP.TF.6.14E:
U 1F1E, BEA6,15 :10700      MOV LS[ALTER.ODD] TO WR[0]
U 1F1F, D89A,15 :10701      INC WR[0]
U 1F20, 53A6,15 :10702      MOV WR[0] TO LS[L53]      ;LS = AAAAAAAB
U 1F21, A180,15 :10703      MOV LS[ALTER.EVEN] TO Q      ;Q = 55555555
U 1F22, 369E,95 :10704      BIS LS[L53] TO Q
U 1F23, 0869,3C :10705      MOV Q TO WR[0]
U 1F24, 89F1,C4 :10706      MOV LS[ONES] TO WR[1]      ;EXPECTED = FFFFFFFF
U 1F25, 8A70,1C :10707      JSR [CHECK.RESULT]      ;CHECK LS.OR.Q
U 1F26, B698,15 :10708      JMP [LOOP.TF.6.14E]      ;LOOP ON ERROR IF ENABLED
U 1F27, 2040,15 :10709      JSR [INC.OTHER]
U 1F28, 3EE6,15 :10710      LOOP.TF.6.14F:
U 1F29, D89A,15 :10711      MOV LS[ALTER.ODD] TO WR[0]
U 1F2A, D3E6,15 :10712      INC WR[0]
U 1F2B, A180,15 :10713      MOV WR[0] TO LS[L73]      ;LS = AAAAAAAB
U 1F2C, 369E,95 :10714      MOV LS[ALTER.EVEN] TO Q      ;Q = 55555555
U 1F2D, 0869,3C :10715      BIS LS[L73] TO Q
U 1F2E, 89F2,64 :10716      MOV Q TO WR[0]
U 1F2F, 8A70,1C :10717      MOV LS[ONES] TO WR[1]      ;EXPECTED = FFFFFFFF
U 1F30, B698,15 :10718      JSR [CHECK.RESULT]      ;CHECK LS.OR.Q
U 1F31, 3E10,15 :10719      JMP [LOOP.TF.6.14F]      ;LOOP ON ERROR IF ENABLED
U 1F32, 589E,15 :10720      JSR [INC.OTHER]
U 1F33, 5410,15 :10721      LOOP.TF.6.150:
U 1F34, A180,15 :10722      MOV LS[ALTER.ODD] TO WR[0]
U 1F35, B69A,95 :10723      MOV WR[0] TO LS[T8]
U 1F36, 0869,3C :10724      MOV LS[ONES] TO Q      ;LS = AAAAAAAA
U 1F37, 09F3,04 :10725      BIC LS[T8] TO Q      ;Q = FFFFFFFF
U 1F38, 8A70,1C :10726      MOV Q TO WR[0]
U 1F39, B698,15 :10727      MOV LS[ALTER.EVEN] TO WR[1]      ;EXPECTED = 55555555
U 1F3A, BE60,15 :10728      JSR [CHECK.RESULT]      ;CHECK LS.MASK.Q
U 1F3B, 589E,15 :10729      JMP [LOOP.TF.6.150]      ;LOOP ON ERROR IF ENABLED
U 1F3C, D460,15 :10730      JSR [INC.OTHER]
U 1F3D, A180,15 :10731      LOOP.TF.6.151:
U 1F3E, B69A,95 :10732      MOV LS[ALTER.ODD] TO WR[0]
U 1F3F, BE60,15 :10733      MOV WR[0] TO LS[L30]      ;LS = AAAAAAAA
U 1F40, 589E,15 :10734      MOV LS[ONES] TO Q      ;Q = FFFFFFFF
U 1F41, D460,15 :10735      BIC LS[L30] TO Q
U 1F42, A180,15 :10736      MOV Q TO WR[0]
U 1F43, B69A,95 :10737      MOV LS[ALTER.EVEN] TO WR[1]      ;EXPECTED = 55555555
  
```

000

XX

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040 1

XXXXXXXXXXXXXXXXXXXXXXXXXXXX

[illegible]


```

U 210C. 0869.3C :11123 JSR [CHECK.RESULT] ;CHECK WRA.FROM.LS.WRB
U 210D. 0A10.74 :11124 JMP [LOOP.TF.6.175] ;LOOP ON ERROR IF ENABLED
U 210E. 8A70.1C :11125 'SR [INC.OTHER]
:11126 LOOP.TF.6.176:
U 210F. 3642.15 :11127 MOV LS[#2] TO WR[0] ;WRO = 2
U 2110. 3641.15 :11128 MOV LS[#1] TO WR[2]
U 2111. 3EA7.15 :11129 MOV WR[2] TO LS[L53] ;LS = 1
U 2112. DDA6.15 :11130 SUB WR[0] FROM LS[L53] TO WR
U 2113. 369E.95 :11131 MOV LS[ONES] TO WR[1] ;EXPECTED = -1
U 2114. 0869.3C :11132 JSR [CHECK.RESULT] ;CHECK WRA.FROM.LS.WRB
U 2115. 8A10.F4 :11133 JMP [LOOP.TF.6.176] ;LOOP ON ERROR IF ENABLED
U 2116. 8A70.1C :11134 JSR [INC.OTHER]
:11135 LOOP.TF.6.177:
U 2117. 3642.15 :11136 MOV LS[#2] TO WR[0] ;WRO = 2
U 2118. 3641.15 :11137 MOV LS[#1] TO WR[2]
U 2119. BEE7.15 :11138 MOV WR[2] TO LS[L73] ;LS = 1
U 211A. 5DE6.15 :11139 SUB WR[0] FROM LS[L73] TO WR
U 211B. 369E.95 :11140 MOV LS[ONES] TO WR[1] ;EXPECTED = -1
U 211C. 0869.3C :11141 JSR [CHECK.RESULT] ;CHECK WRA.FROM.LS.WRB
U 211D. 8A11.74 :11142 JMP [LOOP.TF.6.177] ;LOOP ON ERROR IF ENABLED
U 211E. 8A70.1C :11143 JSR [INC.OTHER]
:11144 LOOP.TF.6.178:
U 211F. B640.15 :11145 MOV LS[#1] TO WR[0]
U 2120. 3E10.15 :11146 MOV WR[0] TO LS[L8] ;LS = 1
U 2121. 5E10.15 :11147 MNEG LS[L8] TO WR[0]
U 2122. 369E.95 :11148 MOV LS[ONES] TO WR[1] ;EXPECTED = -1
U 2123. 0869.3C :11149 JSR [CHECK.RESULT] ;CHECK LS.NEG.WR
U 2124. 0A11.F4 :11150 JMP [LOOP.TF.6.178] ;LOOP ON ERROR IF ENABLED
U 2125. 8A70.1C :11151 JSR [INC.OTHER]
:11152 LOOP.TF.6.179:
U 2126. B640.15 :11153 MOV LS[#1] TO WR[0]
U 2127. BE60.15 :11154 MOV WR[0] TO LS[L30] ;LS = 1
U 2128. DE60.15 :11155 MNEG LS[L30] TO WR[0]
U 2129. 369E.95 :11156 MOV LS[ONES] TO WR[1] ;EXPECTED = -1
U 212A. 0869.3C :11157 JSR [CHECK.RESULT] ;CHECK LS.NEG.WR
U 212B. 0A12.64 :11158 JMP [LOOP.TF.6.179] ;LOOP ON ERROR IF ENABLED
U 212C. 8A70.1C :11159 JSR [INC.OTHER]
:11160 LOOP.TF.6.17A:
U 212D. B640.15 :11161 MOV LS[#1] TO WR[0]
U 212E. BEA6.15 :11162 MOV WR[0] TO LS[L53] ;LS = 1
U 212F. DEA6.15 :11163 MNEG LS[L53] TO WR[0]
U 2130. 369E.95 :11164 MOV LS[ONES] TO WR[1] ;EXPECTED = -1
U 2131. 0869.3C :11165 JSR [CHECK.RESULT] ;CHECK LS.NEG.WR
U 2132. 8A12.D4 :11166 JMP [LOOP.TF.6.17A] ;LOOP ON ERROR IF ENABLED
U 2133. 8A70.1C :11167 JSR [INC.OTHER]
:11168 LOOP.TF.6.17B:
U 2134. B640.15 :11169 MOV LS[#1] TO WR[0]
U 2135. 3EE6.15 :11170 MOV WR[0] TO LS[L73] ;LS = 1
U 2136. 5EE6.15 :11171 MNEG LS[L73] TO WR[0]
U 2137. 369E.95 :11172 MOV LS[ONES] TO WR[1] ;EXPECTED = -1
U 2138. 0869.3C :11173 JSR [CHECK.RESULT] ;CHECK LS.NEG.WR
U 2139. 0A13.44 :11174 JMP [LOOP.TF.6.17B] ;LOOP ON ERROR IF ENABLED
U 213A. 8A70.1C :11175 JSR [INC.OTHER]
:11176 LOOP.TF.6.17C:
U 213B. 369A.15 :11177 MOV LS[ALTER.EVEN] TO WR[0]
  
```

```

U 213C, 3E10,15 :11178      MOV WR[0] TO LS[T8]          ;LS = 55555555
U 213D, DF10,15 :11179      MCOM LS[T8] TO WR[0]
U 213E, 3698,95 :11180      MOV LS[ALTER.ODD] TO WR[1]      ;EXPECTED = AAAAAAAAAA
U 213F, 0869,3C :11181      JSR [CHECK.RESULT]             ;CHECK LS.COM.WR
U 2140, 0A13,B4 :11182      JMP [LOOP.TF.6.17C]             ;LOOP ON ERROR IF ENABLED
U 2141, 8A70,1C :11183      JSR [INC.OTHER]
                        :11184
LOOP.TF.6.17D:
U 2142, 369A,15 :11185      MOV LS[ALTER.EVEN] TO WR[0]
U 2143, BE60,15 :11186      MOV WR[0] TO LS[L30]          ;LS = 55555555
U 2144, 5F60,15 :11187      MCOM LS[L30] TO WR[0]
U 2145, 3698,95 :11188      MOV LS[ALTER.ODD] TO WR[1]      ;EXPECTED = AAAAAAAAAA
U 2146, 0869,3C :11189      JSR [CHECK.RESULT]             ;CHECK LS.COM.WR
U 2147, 8A14,24 :11190      JMP [LOOP.TF.6.17D]             ;LOOP ON ERROR IF ENABLED
U 2148, 8A70,1C :11191      JSR [INC.OTHER]
                        :11192
LOOP.TF.6.17E:
U 2149, 369A,15 :11193      MOV LS[ALTER.EVEN] TO WR[0]
U 214A, BEA6,15 :11194      MOV WR[0] TO LS[L53]          ;LS = 55555555
U 214B, 5FA6,15 :11195      MCOM LS[L53] TO WR[0]
U 214C, 3698,95 :11196      MOV LS[ALTER.ODD] TO WR[1]      ;EXPECTED = AAAAAAAAAA
U 214D, 0869,3C :11197      JSR [CHECK.RESULT]             ;CHECK LS.COM.WR
U 214E, 0A14,94 :11198      JMP [LOOP.TF.6.17E]             ;LOOP ON ERROR IF ENABLED
U 214F, 8A70,1C :11199      JSR [INC.OTHER]
                        :11200
LOOP.TF.6.17F:
U 2150, 369A,15 :11201      MOV LS[ALTER.EVEN] TO WR[0]
U 2151, 3EE6,15 :11202      MOV WR[0] TO LS[L73]          ;LS = 55555555
U 2152, DFE6,15 :11203      MCOM LS[L73] TO WR[0]
U 2153, 3698,95 :11204      MOV LS[ALTER.ODD] TO WR[1]      ;EXPECTED = AAAAAAAAAA
U 2154, 0869,3C :11205      JSR [CHECK.RESULT]             ;CHECK LS.COM.WR
U 2155, 8A15,04 :11206      JMP [LOOP.TF.6.17F]             ;LOOP ON ERROR IF ENABLED
U 2156, 8A70,1C :11207      JSR [INC.OTHER]
                        :11208
; LS Destination tests
LOOP.TF.6.180:
U 2157, 3642,15 :11210      MOV LS[#2] TO WR[0]
U 2158, 3E10,15 :11211      MOV WR[0] TO LS[T8]          ;LS = 2
U 2159, B69E,15 :11212      MOV LS[ONES] TO WR[0]          ;WRO = -1
                        :11213
                        ADD LS[T8] TO WR[0],
U 215A, E010,15 :11214      XCHG
U 215B, 3640,95 :11215      MOV LS[#1] TO WR[1]          ;EXPECTED = 1
U 215C, 0869,3C :11216      JSR [CHECK.RESULT]             ;CHECK LS.PLUS.WR.XCHG
U 215D, 0A15,74 :11217      JMP [LOOP.TF.6.180]             ;LOOP ON ERROR IF ENABLED
U 215E, B610,15 :11218      MOV LS[T8] TO WR[0]
U 215F, 369E,95 :11219      MOV LS[ONES] TO WR[1]          ;EXPECTED = -1
U 2160, 0869,3C :11220      JSR [CHECK.RESULT]             ;CHECK XCHNG
U 2161, 0A15,74 :11221      JMP [LOOP.TF.6.180]             ;LOOP ON ERROR IF ENABLED
U 2162, 8A70,1C :11222      JSR [INC.OTHER]
                        :11223
LOOP.TF.6.181:
U 2163, 3642,15 :11224      MOV LS[#2] TO WR[0]
U 2164, BE60,15 :11225      MOV WR[0] TO LS[L30]          ;LS = 2
U 2165, B69E,15 :11226      MOV LS[ONES] TO WR[0]          ;WRO = -1
                        :11227
                        ADD LS[L30] TO WR[0],
U 2166, 6060,15 :11228      XCHG
U 2167, 3640,95 :11229      MOV LS[#1] TO WR[1]          ;EXPECTED = 1
U 2168, 0869,3C :11230      JSR [CHECK.RESULT]             ;CHECK LS.PLUS.WR.XCHG
U 2169, 8A16,34 :11231      JMP [LOOP.TF.6.181]             ;LOOP ON ERROR IF ENABLED
U 216A, 3660,15 :11232      MOV LS[L30] TO WR[0]
  
```

```

U 216B, 369E,95 :11233      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 216C, 0869,3C :11234      JSR [CHECK.RESULT]      ;CHECK XCHNG
U 216D, 8A16,34 :11235      JMP [LOOP.TF.6.181]      ;LOOP ON ERROR IF ENABLED
U 216E, 8A70,1C :11236      JSR [INC.OTHER]
                                LOOP.TF.6.182:
U 216F, 3642,15 :11238      MOV LS[#2] TO WR[0]
U 2170, BEA6,15 :11239      MOV WR[0] TO LS[L53]      ;LS = 2
U 2171, B69E,15 :11240      MOV LS[ONES] TO WR[0]      ;WRO = -1
                                :11241
                                ADD LS[L53] TO WR[0],
                                XCHG
U 2172, 60A6,15 :11242      MOV LS[#1] TO WR[1]      ;EXPECTED = 1
U 2173, 3640,95 :11243      JSR [CHECK.RESULT]      ;CHECK LS.PLUS.WR.XCHG
U 2174, 0869,3C :11244      JMP [LOOP.TF.6.182]      ;LOOP ON ERROR IF ENABLED
U 2175, 8A16,F4 :11245      MOV LS[L53] TO WR[0]
U 2176, 36A6,15 :11246      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 2177, 369E,95 :11247      JSR [CHECK.RESULT]      ;CHECK XCHNG
U 2178, 0869,3C :11248      JMP [LOOP.TF.6.182]      ;LOOP ON ERROR IF ENABLED
U 2179, 8A16,F4 :11249      JSR [INC.OTHER]
                                LOOP.TF.6.183:
U 217B, 3642,15 :11252      MOV LS[#2] TO WR[0]
U 217C, 3EE6,15 :11253      MOV WR[0] TO LS[L73]      ;LS = 2
U 217D, B69E,15 :11254      MOV LS[ONES] TO WR[0]      ;WRO = -1
                                :11255
                                ADD LS[L73] TO WR[0],
                                XCHG
U 217E, E0E6,15 :11256      MOV LS[#1] TO WR[1]      ;EXPECTED = 1
U 217F, 3640,95 :11257      JSR [CHECK.RESULT]      ;CHECK LS.PLUS.WR.XCHG
U 2180, 0869,3C :11258      JMP [LOOP.TF.6.183]      ;LOOP ON ERROR IF ENABLED
U 2181, 8A17,B4 :11259      MOV LS[L73] TO WR[0]
U 2182, B6E6,15 :11260      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 2183, 369E,95 :11261      JSR [CHECK.RESULT]      ;CHECK XCHNG
U 2184, 0869,3C :11262      JMP [LOOP.TF.6.183]      ;LOOP ON ERROR IF ENABLED
U 2185, 8A17,B4 :11263      JSR [INC.OTHER]
                                LOOP.TF.6.184:
U 2187, 3642,15 :11266      MOV LS[#2] TO WR[0]
U 2188, 3E10,15 :11267      MOV WR[0] TO LS[T8]      ;LS = 2
U 2189, B640,15 :11268      MOV LS[#1] TO WR[0]      ;WRO = 1
                                :11269
                                SUB LS[T8] FROM WR[0],
                                XCHG
U 218A, 6110,15 :11270      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 218B, 369E,95 :11271      JSR [CHECK.RESULT]      ;CHECK LS.FROM.WR.XCHG
U 218C, 0869,3C :11272      JMP [LOOP.TF.6.184]      ;LOOP ON ERROR IF ENABLED
U 218D, 8A18,74 :11273      MOV LS[T8] TO WR[0]
U 218E, B610,15 :11274      MOV LS[#1] TO WR[1]      ;EXPECTED = 1
U 218F, 3640,95 :11275      JSR [CHECK.RESULT]      ;CHECK XCHG
U 2190, 0869,3C :11276      JMP [LOOP.TF.6.184]      ;LOOP ON ERROR IF ENABLED
U 2191, 8A18,74 :11277      JSR [INC.OTHER]
                                LOOP.TF.6.185:
U 2193, 3642,15 :11280      MOV LS[#2] TO WR[0]
U 2194, BE60,15 :11281      MOV WR[0] TO LS[L30]      ;LS = 2
U 2195, B640,15 :11282      MOV LS[#1] TO WR[0]      ;WRO = 1
                                :11283
                                SUB LS[L30] FROM WR[0],
                                XCHG
U 2196, E160,15 :11284      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 2197, 369E,95 :11285      JSR [CHECK.RESULT]      ;CHECK LS.FROM.WR.XCHG
U 2198, 0869,3C :11286      JMP [LOOP.TF.6.185]      ;LOOP ON ERROR IF ENABLED
U 2199, 8A19,34 :11287
  
```

XX

```

U 21C9, 4518,95 :11343      BIC LS[T12] TO WR[1]      ;EXPECTED = 0000AAAA
U 21CA, 0869,3C :11344      JSR [CHECK.RESULT]      ;CHECK LS.AND.WR.XCHG
U 21CB, 0A1C,44 :11345      JMP [LOOP.TF.6.189]      ;LOOP ON ERROR IF ENABLED
U 21CC, 3660,15 :11346      MOV LS[L30] TO WR[0]
U 21CD, B628,95 :11347      MOV LS[FFFF] TO WR[1]      ;EXPECTED = 0000FFFF
U 21CE, 0869,3C :11348      JSR [CHECK.RESULT]      ;CHECK XCHG
U 21CF, 0A1C,44 :11349      JMP [LOOP.TF.6.189]      ;LOOP ON ERROR IF ENABLED
U 21D0, 8A70,1C :11350      JSR [INC.OTHER]
:11351
LOOP.TF.6.18A:
U 21D1, B698,15 :11352      MOV LS[ALTER.ODD] TO WR[0]
U 21D2, BEA6,15 :11353      MOV WR[0] TO LS[L53]      ; LS = AAAAAAAA
U 21D3, 3628,15 :11354      MOV LS[FFFF] TO WR[0]      ;WRO = 0000FFFF
:11355
AND LS[L53] TO WR[0],
XCHG
U 21D4, E2A6,15 :11356      MOV LS[ALTER.ODD] TO WR[1]
U 21D5, 3698,95 :11357      BIC LS[T12] TO WR[1]      ;EXPECTED = 0000AAAA
U 21D6, 4518,95 :11358      JSR [CHECK.RESULT]      ;CHECK LS.AND.WR.XCHG
U 21D7, 0869,3C :11359      JMP [LOOP.TF.6.18A]      ;LOOP ON ERROR IF ENABLED
U 21D8, 8A1D,14 :11360      MOV LS[L53] TO WR[0]
U 21D9, 36A6,15 :11361      MOV LS[FFFF] TO WR[1]      ;EXPECTED = 0000FFFF
U 21DA, B628,95 :11362      JSR [CHECK.RESULT]      ;CHECK XCHG
U 21DB, 0869,3C :11363      JMP [LOOP.TF.6.18A]      ;LOOP ON ERROR IF ENABLED
U 21DC, 8A1D,14 :11364      JSR [INC.OTHER]
:11365
LOOP.TF.6.18B:
U 21DE, B698,15 :11367      MOV LS[ALTER.ODD] TO WR[0]
U 21DF, 3EE6,15 :11368      MOV WR[0] TO LS[L73]      ; LS = AAAAAAAA
U 21E0, 3628,15 :11369      MOV LS[FFFF] TO WR[0]      ;WRO = 0000FFFF
:11370
AND LS[L73] TO WR[0],
XCHG
U 21E1, 62E6,15 :11371      MOV LS[ALTER.ODD] TO WR[1]
U 21E2, 3698,95 :11372      BIC LS[T12] TO WR[1]      ;EXPECTED = 0000AAAA
U 21E3, 4518,95 :11373      JSR [CHECK.RESULT]      ;CHECK LS.AND.WR.XCHG
U 21E4, 0869,3C :11374      JMP [LOOP.TF.6.18B]      ;LOOP ON ERROR IF ENABLED
U 21E5, 8A1D,E4 :11375      MOV LS[L73] TO WR[0]
U 21E6, B6E6,15 :11376      MOV LS[FFFF] TO WR[1]      ;EXPECTED = 0000FFFF
U 21E7, B628,95 :11377      JSR [CHECK.RESULT]      ;CHECK XCHG
U 21E8, 0869,3C :11378      JMP [LOOP.TF.6.18B]      ;LOOP ON ERROR IF ENABLED
U 21E9, 8A1D,E4 :11379      JSR [INC.OTHER]
:11380
LOOP.TF.6.18C:
U 21EB, B698,15 :11381      MOV LS[ALTER.ODD] TO WR[0]
U 21EC, 3E10,15 :11382      MOV WR[0] TO LS[T8]      ; LS = AAAAAAAA
U 21ED, 3628,15 :11383      MOV LS[FFFF] TO WR[0]      ;WRO = 0000FFFF
:11384
XOR LS[T8] TO WR[0],
XCHG
U 21EE, E310,15 :11386      MOV LS[ALTER.MIX] TO WR[1]
U 21EF, 3722,95 :11387      JSR [CHECK.RESULT]      ;EXPECTED = AAAA5555
U 21F0, 0869,3C :11388      JMP [LOOP.TF.6.18C]      ;CHECK LS.XOR.WR.XCHG
U 21F1, 8A1E,B4 :11389      MOV LS[T8] TO WR[0]      ;LOOP ON ERROR IF ENABLED
U 21F2, B610,15 :11390      MOV LS[FFFF] TO WR[1]
U 21F3, B628,95 :11391      JSR [CHECK.RESULT]      ;EXPECTED = 0000FFFF
U 21F4, 0869,3C :11392      JMP [LOOP.TF.6.18C]      ;CHECK XCHG
U 21F5, 8A1E,B4 :11393      JSR [INC.OTHER]      ;LOOP ON ERROR IF ENABLED
:11394
LOOP.TF.6.18D:
U 21F7, B698,15 :11395      MOV LS[ALTER.ODD] TO WR[0]
U 21F8, BE60,15 :11397      MOV WR[0] TO LS[L30]      ; LS = AAAAAAAA
    
```



```

U 21F9, 3628,15 :11398 MOV LS[#####] TO WR[0] ;WRO = 0000FFFF
                  :11399 XOR LS[L30] TO WR[0],
U 21FA, 6360,15 :11400 XCHG
U 21FB, 3722,95 :11401 MOV LS[ALTER.MIX] TO WR[1] ;EXPECTED = AAAA5555
U 21FC, 0869,3C :11402 JSR [CHECK.RESULT] ;CHECK LS.XOR.WR.XCHG
U 21FD, 0A1F,74 :11403 JMP [LOOP.TF.6.18D] ;LOOP ON ERROR IF ENABLED
U 21FE, 3660,15 :11404 MOV LS[L30] TO WR[0]
U 21FF, B628,95 :11405 MOV LS[#####] TO WR[1] ;EXPECTED = 0000FFFF
U 2200, 0869,3C :11406 JSR [CHECK.RESULT] ;CHECK XCHG
U 2201, 0A1F,74 :11407 JMP [LOOP.TF.6.18D] ;LOOP ON ERROR IF ENABLED
U 2202, 8A70,1C :11408 JSR [INC.OTHER]
                  :11409
U 2203, B698,15 :11410 LOOP.TF.6.18E: MOV LS[ALTER.ODD] TO WR[0]
U 2204, BEA6,15 :11411 MOV WR[0] TO LS[L53] ;LS = AAAAAAAA
U 2205, 3628,15 :11412 MOV LS[#####] TO WR[0] ;WRO = 0000FFFF
                  :11413 XOR LS[L53] TO WR[0],
U 2206, 63A6,15 :11414 XCHG
U 2207, 3722,95 :11415 MOV LS[ALTER.MIX] TO WR[1] ;EXPECTED = AAAA5555
U 2208, 0869,3C :11416 JSR [CHECK.RESULT] ;CHECK LS.XOR.WR.XCHG
U 2209, 8A20,34 :11417 JMP [LOOP.TF.6.18E] ;LOOP ON ERROR IF ENABLED
U 220A, 36A6,15 :11418 MOV LS[L53] TO WR[0]
U 220B, B628,95 :11419 MOV LS[#####] TO WR[1] ;EXPECTED = 0000FFFF
U 220C, 0869,3C :11420 JSR [CHECK.RESULT] ;CHECK XCHG
U 220D, 8A20,34 :11421 JMP [LOOP.TF.6.18E] ;LOOP ON ERROR IF ENABLED
U 220E, 8A70,1C :11422 JSR [INC.OTHER]
                  :11423
U 220F, B698,15 :11424 LOOP.TF.6.18F: MOV LS[ALTER.ODD] TO WR[0]
U 2210, 3EE6,15 :11425 MOV WR[0] TO LS[L73] ;LS = AAAAAAAA
U 2211, 3628,15 :11426 MOV LS[#####] TO WR[0] ;WRO = 0000FFFF
                  :11427 XOR LS[L73] TO WR[0],
U 2212, E3E6,15 :11428 XCHG
U 2213, 3722,95 :11429 MOV LS[ALTER.MIX] TO WR[1] ;EXPECTED = AAAA5555
U 2214, 0869,3C :11430 JSR [CHECK.RESULT] ;CHECK LS.XOR.WR.XCHG
U 2215, 8A20,F4 :11431 JMP [LOOP.TF.6.18F] ;LOOP ON ERROR IF ENABLED
U 2216, B6E6,15 :11432 MOV LS[L73] TO WR[0]
U 2217, B628,95 :11433 MOV LS[#####] TO WR[1] ;EXPECTED = 0000FFFF
U 2218, 0869,3C :11434 JSR [CHECK.RESULT] ;CHECK XCHG
U 2219, 8A20,F4 :11435 JMP [LOOP.TF.6.18F] ;LOOP ON ERROR IF ENABLED
U 221A, 8A70,1C :11436 JSR [INC.OTHER]
                  :11437
U 221B, B698,15 :11438 LOOP.TF.6.190: MOV LS[ALTER.ODD] TO WR[0]
U 221C, 3E10,15 :11439 MOV WR[0] TO LS[T8] ;LS = AAAAAA
U 221D, 369A,15 :11440 MOV LS[ALTER.EVEN] TO WR[0] ;WRO = 555555
U 221E, 6410,15 :11441 SWAP LS[T8] WITH WR[0]
U 221F, 3698,95 :11442 MOV LS[ALTER.ODD] TO WR[1] ;EXPECTED IN WR = AAAAAAAA
U 2220, 0869,3C :11443 JSR [CHECK.RESULT] ;CHECK SWAP.LS.WR
U 2221, 8A21,B4 :11444 JMP [LOOP.TF.6.190] ;LOOP ON ERROR IF ENABLED
U 2222, B610,15 :11445 MOV LS[T8] TO WR[0]
U 2223, B69A,95 :11446 MOV LS[ALTER.EVEN] TO WR[1] ;EXPECTED IN LS = 55555555
U 2224, 0869,3C :11447 JSR [CHECK.RESULT] ;CHECK SWAP.LS.WR
U 2225, 8A21,B4 :11448 JMP [LOOP.TF.6.190] ;LOOP ON ERROR IF ENABLED
U 2226, 8A70,1C :11449 JSR [INC.OTHER]
                  :11450
U 2227, B698,15 :11451 LOOP.TF.6.191: MOV LS[ALTER.ODD] TO WR[0]
U 2228, BE60,15 :11452 MOV WR[0] TO LS[L30] ;LS = AAAAAA

```

[illegible]

```

U 2229, 369A,15 :11453      MOV LS[ALTER.EVEN] TO WR[0]      ;WRO = 5555555
U 222A, E460,15 :11454      SWAP LS[L30] WITH WR[0]
U 222B, 3698,95 :11455      MOV LS[ALTER.ODD] TO WR[1]      ;EXPECTED IN WR = AAAAAAAA
U 222C, 0869,3C :11456      JSR [CHECK.RESULT]              ;CHECK SWAP.LS.WR
U 222D, 8A22,74 :11457      JMP [LOOP.TF.6.191]              ;LOOP ON ERROR IF ENABLED
U 222E, 3660,15 :11458      MOV LS[L30] TO WR[0]
U 222F, B69A,95 :11459      MOV LS[ALTER.EVEN] TO WR[1]      ;EXPECTED IN LS = 5555555
U 2230, 0869,3C :11460      JSR [CHECK.RESULT]              ;CHECK SWAP.LS.WR
U 2231, 8A22,74 :11461      JMP [LOOP.TF.6.191]              ;LOOP ON ERROR IF ENABLED
U 2232, 8A70,1C :11462      JSR [INC.OTHER]
      LOOP.TF.6.192:
U 2233, B698,15 :11463      MOV LS[ALTER.ODD] TO WR[0]
U 2234, BEA6,15 :11464      MOV WR[0] TO LS[L53]
U 2235, 369A,15 :11465      MOV LS[ALTER.EVEN] TO WR[0]      ;LS = AAAAAAA
U 2236, E4A6,15 :11466      SWAP LS[L53] WITH WR[0]          ;WRO = 5555555
U 2237, 3698,95 :11467      MOV LS[ALTER.ODD] TO WR[1]      ;EXPECTED IN WR = AAAAAAAA
U 2238, 0869,3C :11468      JSR [CHECK.RESULT]              ;CHECK SWAP.LS.WR
U 2239, 8A23,34 :11469      JMP [LOOP.TF.6.192]              ;LOOP ON ERROR IF ENABLED
U 223A, 36A6,15 :11470      MOV LS[L53] TO WR[0]
U 223B, B69A,95 :11471      MOV LS[ALTER.EVEN] TO WR[1]      ;EXPECTED IN LS = 5555555
U 223C, 0869,3C :11472      JSR [CHECK.RESULT]              ;CHECK SWAP.LS.WR
U 223D, 8A23,34 :11473      JMP [LOOP.TF.6.192]              ;LOOP ON ERROR IF ENABLED
U 223E, 8A70,1C :11474      JSR [INC.OTHER]
      LOOP.TF.6.193:
U 223F, B698,15 :11475      MOV LS[ALTER.ODD] TO WR[0]
U 2240, 3EE6,15 :11476      MOV WR[0] TO LS[L73]
U 2241, 369A,15 :11477      MOV LS[ALTER.EVEN] TO WR[0]      ;LS = AAAAAAA
U 2242, 64E6,15 :11478      SWAP LS[L73] WITH WR[0]          ;WRO = 5555555
U 2243, 3698,95 :11479      MOV LS[ALTER.ODD] TO WR[1]      ;EXPECTED IN WR = AAAAAAAA
U 2244, 0869,3C :11480      JSR [CHECK.RESULT]              ;CHECK SWAP.LS.WR
U 2245, 8A23,F4 :11481      JMP [LOOP.TF.6.193]              ;LOOP ON ERROR IF ENABLED
U 2246, B6E6,15 :11482      MOV LS[L73] TO WR[0]
U 2247, B69A,95 :11483      MOV LS[ALTER.EVEN] TO WR[1]      ;EXPECTED IN LS = 5555555
U 2248, 0869,3C :11484      JSR [CHECK.RESULT]              ;CHECK SWAP.LS.WR
U 2249, 8A23,F4 :11485      JMP [LOOP.TF.6.193]              ;LOOP ON ERROR IF ENABLED
U 224A, 8A70,1C :11486      JSR [INC.OTHER]
      LOOP.TF.6.194:
U 224B, 369A,15 :11487      MOV LS[ALTER.EVEN] TO WR[0]
U 224C, 3E10,15 :11488      MOV WR[0] TO LS[T8]
U 224D, E510,15 :11489      CLR LS[T8]
U 224E, B610,15 :11490      MOV LS[T8] TO WR[0]
U 224F, 2F82,95 :11491      CLR WR[1]
U 2250, 0869,3C :11492      JSR [CHECK.RESULT]              ;EXPECTED = 0
U 2251, 8A24,B4 :11493      JMP [LOOP.TF.6.194]              ;CHECK CLR.LS
U 2252, 8A70,1C :11494      JSR [INC.OTHER]              ;LOOP ON ERROR IF ENABLED
      LOOP.TF.6.195:
U 2253, 369A,15 :11495      MOV LS[ALTER.EVEN] TO WR[0]
U 2254, BE60,15 :11496      MOV WR[0] TO LS[L30]
U 2255, 6560,15 :11497      CLR LS[L30]
U 2256, 3660,15 :11498      MOV LS[L30] TO WR[0]
U 2257, 2F82,95 :11499      CLR WR[1]
U 2258, 0869,3C :11500      JSR [CHECK.RESULT]              ;EXPECTED = 0
U 2259, 8A25,34 :11501      JMP [LOOP.TF.6.195]              ;CHECK CLR.LS
U 225A, 8A70,1C :11502      JSR [INC.OTHER]              ;LOOP ON ERROR IF ENABLED
      LOOP.TF.6.196:
  
```

XXXXXXXXXXXXXXXXXXXX

XX

[illegible]

ZZ-ENKCB-1.0
: ENKCB.MCR
: ENKCB.MIC

: ENKCB.MIC

MICRO2 1L(03) 3-MAR-82 10:02:46
TEST F - DEST CTL ROM And SRC.ALU CTL PAL Tests (M8390 CPU Module)

Fiche 2 Frame H4

Sequence 252
Rev 01.00

Page 246

```
U 22F1, 3E61,15 :11673      MOV WR[2] TO LS[L30]          ; LS = 2
U 22F2, 3641,15 :11674      MOV LS[#1] TO WR[2]          ; WR2 = 1
U 22F3, E961,15 :11675      SUB LS[L30] FROM WR[2] TO LS
U 22F4, 3660,15 :11676      MOV LS[L30] TO WR[0]
U 22F5, 369E,95 :11677      MOV LS[ONES] TO WR[1]          ; EXPECTED = -1
U 22F6, 0869,3C :11678      JSR [CHECK.RESULT]          ; CHECK LS.FROM.WR.LS
U 22F7, 8A2F,04 :11679      JMP [LOOP.TF.6.1A5]          ; LOOP ON ERROR IF ENABLED
U 22F8, 8A70,1C :11680      JSR [INC.OTHER]
:11681      LOOP.TF.6.1A6:
U 22F9, B643,15 :11682      MOV LS[#2] TO WR[2]
U 22FA, 3EA7,15 :11683      MOV WR[2] TO LS[L53]          ; LS = 2
U 22FB, 3641,15 :11684      MOV LS[#1] TO WR[2]          ; WR2 = 1
U 22FC, E9A7,15 :11685      SUB LS[L53] FROM WR[2] TO LS
U 22FD, 36A6,15 :11686      MOV LS[L53] TO WR[0]
U 22FE, 369E,95 :11687      MOV LS[ONES] TO WR[1]          ; EXPECTED = -1
U 22FF, 0869,3C :11688      JSR [CHECK.RESULT]          ; CHECK LS.FROM.WR.LS
U 2300, 8A2F,94 :11689      JMP [LOOP.TF.6.1A6]          ; LOOP ON ERROR IF ENABLED
U 2301, 8A70,1C :11690      JSR [INC.OTHER]
:11691      LOOP.TF.6.1A7:
U 2302, B643,15 :11692      MOV LS[#2] TO WR[2]
U 2303, BEE7,15 :11693      MOV WR[2] TO LS[L73]          ; LS = 2
U 2304, 3641,15 :11694      MOV LS[#1] TO WR[2]          ; WR2 = 1
U 2305, 69E7,15 :11695      SUB LS[L73] FROM WR[2] TO LS
U 2306, B6E6,15 :11696      MOV LS[L73] TO WR[0]
U 2307, 369E,95 :11697      MOV LS[ONES] TO WR[1]          ; EXPECTED = -1
U 2308, 0869,3C :11698      JSR [CHECK.RESULT]          ; CHECK LS.FROM.WR.LS
U 2309, 8A30,24 :11699      JMP [LOOP.TF.6.1A7]          ; LOOP ON ERROR IF ENABLED
U 230A, 8A70,1C :11700      JSR [INC.OTHER]
:11701      LOOP.TF.6.1A8:
U 230B, 369F,15 :11702      MOV LS[ONES] TO WR[2]          ; WR2 = -1
U 230C, B640,15 :11703      MOV LS[#1] TO WR[0]
U 230D, 3E10,15 :11704      MOV WR[0] TO LS[T8]
U 230E, 6A11,15 :11705      ADD (WR[2] TO LS[T8])+1          ; LS = 1
U 230F, B610,15 :11706      MOV LS[T8] TO WR[0]
U 2310, 3640,95 :11707      MOV LS[#1] TO WR[1]          ; EXPECTED = 1
U 2311, 0869,3C :11708      JSR [CHECK.RESULT]          ; CHECK WR.PLUS.LS+1
U 2312, 8A30,B4 :11709      JMP [LOOP.TF.6.1A8]          ; LOOP ON ERROR IF ENABLED
U 2313, 8A70,1C :11710      JSR [INC.OTHER]
:11711      LOOP.TF.6.1A9:
U 2314, 369F,15 :11712      MOV LS[ONES] TO WR[2]          ; WR2 = -1
U 2315, B640,15 :11713      MOV LS[#1] TO WR[0]
U 2316, BE60,15 :11714      MOV WR[0] TO LS[L30]          ; LS = 1
U 2317, EA61,15 :11715      ADD (WR[2] TO LS[L30])+1
U 2318, 3660,15 :11716      MOV LS[L30] TO WR[0]
U 2319, 3640,95 :11717      MOV LS[#1] TO WR[1]          ; EXPECTED = 1
U 231A, 0869,3C :11718      JSR [CHECK.RESULT]          ; CHECK WR.PLUS.LS+1
U 231B, 0A31,44 :11719      JMP [LOOP.TF.6.1A9]          ; LOOP ON ERROR IF ENABLED
U 231C, 8A70,1C :11720      JSR [INC.OTHER]
:11721      LOOP.TF.6.1AA:
U 231D, 369F,15 :11722      MOV LS[ONES] TO WR[2]          ; WR2 = -1
U 231E, B640,15 :11723      MOV LS[#1] TO WR[0]
U 231F, BEA6,15 :11724      MOV WR[0] TO LS[L53]          ; LS = 1
U 2320, FAA7,15 :11725      ADD (WR[2] TO LS[L53])+1
U 2321, 36A6,15 :11726      MOV LS[L53] TO WR[0]
U 2322, 3640,95 :11727      MOV LS[#1] TO WR[1]          ; EXPECTED = 1
```

```

U 2323. 0869.3C :11728      JSR [CHECK.RESULT]          ;CHECK WR.PLUS.LS+1
U 2324. 0A31.D4 :11729      JMP [LOOP.TF.6.1AA]          ;LOOP ON ERROR IF ENABLED
U 2325. 8A70.1C :11730      JSR [INC.OTHER]
U 2326. 369F.15 :11731      LOOP.TF.6.1AB:
U 2327. B640.15 :11732      MOV LS[ONES] TO WR[2]          ;WR2 = -1
U 2328. 3EE6.15 :11733      MOV LS[#1] TO WR[0]
U 2329. 6AE7.15 :11734      MOV WR[0] TO LS[L73]          ;LS = 1
U 232A. B6E6.15 :11735      ADD (WR[2] TO LS[L73])+1
U 232B. 3640.95 :11736      MOV LS[L73] TO WR[0]
U 232C. 0869.3C :11737      MOV LS[#1] TO WR[1]          ;EXPECTED = 1
U 232D. 8A32.64 :11738      JSR [CHECK.RESULT]          ;CHECK WR.PLUS.LS+1
U 232E. 8A70.1C :11739      JMP [LOOP.TF.6.1AB]          ;LOOP ON ERROR IF ENABLED
U 232F. B643.15 :11740      JSR [INC.OTHER]
U 2330. B640.15 :11741      LOOP.TF.6.1AC:
U 2331. 3E10.15 :11742      MOV LS[#2] TO WR[2]          ;WR2 = 2
U 2332. EB11.15 :11743      MOV LS[#1] TO WR[0]
U 2333. B610.15 :11744      MOV WR[0] TO LS[T8]          ;LS = 1
U 2334. 369E.95 :11745      SUB WR[2] FROM LS[T8]
U 2335. 0869.3C :11746      MOV LS[T8] TO WR[0]
U 2336. 8A32.F4 :11747      MOV LS[ONES] TO WR[1]          ;EXPECTED = -1
U 2337. 8A70.1C :11748      JSR [CHECK.RESULT]          ;CHECK WR.FROM.LS
U 2338. B643.15 :11749      JMP [LOOP.TF.6.1AC]          ;LOOP ON ERROR IF ENABLED
U 2339. B640.15 :11750      JSR [INC.OTHER]
U 233A. BE60.15 :11751      LOOP.TF.6.1AD:
U 233B. 6B61.15 :11752      MOV LS[#2] TO WR[2]          ;WR2 = 2
U 233C. 3660.15 :11753      MOV LS[#1] TO WR[0]
U 233D. 369E.95 :11754      MOV WR[0] TO LS[L30]          ;LS = 1
U 233E. 0869.3C :11755      SUB WR[2] FROM LS[L30]
U 233F. 8A33.84 :11756      MOV LS[L30] TO WR[0]
U 2340. 8A70.1C :11757      MOV LS[ONES] TO WR[1]          ;EXPECTED = -1
U 2341. B643.15 :11758      JSR [CHECK.RESULT]          ;CHECK WR.FROM.LS
U 2342. B640.15 :11759      JMP [LOOP.TF.6.1AD]          ;LOOP ON ERROR IF ENABLED
U 2343. BEA6.15 :11760      JSR [INC.OTHER]
U 2344. 6BA7.15 :11761      LOOP.TF.6.1AE:
U 2345. 36A6.15 :11762      MOV LS[#2] TO WR[2]          ;WR2 = 2
U 2346. 369E.95 :11763      MOV LS[#1] TO WR[0]
U 2347. 0869.3C :11764      MOV WR[0] TO LS[L53]          ;LS = 1
U 2348. 0A34.14 :11765      SUB WR[2] FROM LS[L53]
U 2349. 8A70.1C :11766      MOV LS[L53] TO WR[0]
U 234A. B643.15 :11767      MOV LS[ONES] TO WR[1]          ;EXPECTED = -1
U 234B. B640.15 :11768      JSR [CHECK.RESULT]          ;CHECK WR.FROM.LS
U 234C. 3EE6.15 :11769      JMP [LOOP.TF.6.1AE]          ;LOOP ON ERROR IF ENABLED
U 234D. EBE7.15 :11770      JSR [INC.OTHER]
U 234E. B6E6.15 :11771      LOOP.TF.6.1AF:
U 234F. 369E.95 :11772      MOV LS[#2] TO WR[2]          ;WR2 = 2
U 2350. 0869.3C :11773      MOV LS[#1] TO WR[0]
U 2351. 8A34.A4 :11774      MOV WR[0] TO LS[L73]          ;LS = 1
U 2352. 8A70.1C :11775      SUB WR[2] FROM LS[L73]
U 2353. B643.15 :11776      MOV LS[L73] TO WR[0]
U 2354. B640.15 :11777      MOV LS[ONES] TO WR[1]          ;EXPECTED = -1
U 2355. 369E.95 :11778      JSR [CHECK.RESULT]          ;CHECK WR.FROM.LS
U 2356. 8A34.A4 :11779      JMP [LOOP.TF.6.1AF]          ;LOOP ON ERROR IF ENABLED
U 2357. 8A70.1C :11780      JSR [INC.OTHER]
U 2358. B643.15 :11781      LOOP.TF.6.1B0:
U 2359. B640.15 :11782      MOV LS[#2] TO WR[2]          ;WR2 = 2
  
```

```

U 2354. B69E.15 :11783      MOV LS[ONES] TO WR[0]
U 2355. 3E10.15 :11784      MOV WR[0] TO LS[T8]
U 2356. 6C11.15 :11785      ADD WR[2] TO LS[T8]
U 2357. B610.15 :11786      MOV LS[T8] TO WR[0]
U 2358. 3640.95 :11787      MOV LS[#1] TO WR[1]
U 2359. 0869.3C :11788      JSR [CHECK.RESULT]
U 235A. 0A35.34 :11789      JMP [LOOP.TF.6.1B0]
U 235B. 8A70.1C :11790      JSR [INC.OTHER]
                        :11791
LOOP.TF.6.1B1:
U 235C. B643.15 :11792      MOV LS[#2] TO WR[2]
U 235D. B69E.15 :11793      MOV LS[ONES] TO WR[0]
U 235E. BE60.15 :11794      MOV WR[0] TO LS[L30]
U 235F. EC61.15 :11795      ADD WR[2] TO LS[L30]
U 2360. 3660.15 :11796      MOV LS[L30] TO WR[0]
U 2361. 3640.95 :11797      MOV LS[#1] TO WR[1]
U 2362. 0869.3C :11798      JSR [CHECK.RESULT]
U 2363. 0A35.C4 :11799      JMP [LOOP.TF.6.1B1]
U 2364. 8A70.1C :11800      JSR [INC.OTHER]
                        :11801
LOOP.TF.6.1B2:
U 2365. B643.15 :11802      MOV LS[#2] TO WR[2]
U 2366. B69E.15 :11803      MOV LS[ONES] TO WR[0]
U 2367. BEA6.15 :11804      MOV WR[0] TO LS[L53]
U 2368. ECA7.15 :11805      ADD WR[2] TO LS[L53]
U 2369. 36A6.15 :11806      MOV LS[L53] TO WR[0]
U 236A. 3640.95 :11807      MOV LS[#1] TO WR[1]
U 236B. 0869.3C :11808      JSR [CHECK.RESULT]
U 236C. 0A36.54 :11809      JMP [LOOP.TF.6.1B2]
U 236D. 8A70.1C :11810      JSR [INC.OTHER]
                        :11811
LOOP.TF.6.1B3:
U 236E. B643.15 :11812      MOV LS[#2] TO WR[2]
U 236F. B69E.15 :11813      MOV LS[ONES] TO WR[0]
U 2370. 3EE6.15 :11814      MOV WR[0] TO LS[L73]
U 2371. 6CE7.15 :11815      ADD WR[2] TO LS[L73]
U 2372. B6E6.15 :11816      MOV LS[L73] TO WR[0]
U 2373. 3640.95 :11817      MOV LS[#1] TO WR[1]
U 2374. 0869.3C :11818      JSR [CHECK.RESULT]
U 2375. 8A36.E4 :11819      JMP [LOOP.TF.6.1B3]
U 2376. 8A70.1C :11820      JSR [INC.OTHER]
                        :11821
LOOP.TF.6.1B4:
U 2377. 3699.15 :11822      MOV LS[ALTER.ODD] TO WR[2]
U 2378. 2045.15 :11823      INC WR[2]
U 2379. 369A.15 :11824      MOV LS[ALTER.EVEN] TO WR[0]
U 237A. 3E10.15 :11825      MOV WR[0] TO LS[T8]
U 237B. ED11.15 :11826      BIS WR[2] TO LS[T8]
U 237C. B610.15 :11827      MOV LS[T8] TO WR[0]
U 237D. 369E.95 :11828      MOV LS[ONES] TO WR[1]
U 237E. 0869.3C :11829      JSR [CHECK.RESULT]
U 237F. 0A37.74 :11830      JMP [LOOP.TF.6.1B4]
U 2380. 8A70.1C :11831      JSR [INC.OTHER]
                        :11832
LOOP.TF.6.1B5:
U 2381. 3699.15 :11833      MOV LS[ALTER.ODD] TO WR[2]
U 2382. 2045.15 :11834      INC WR[2]
U 2383. 369A.15 :11835      MOV LS[ALTER.EVEN] TO WR[0]
U 2384. BE60.15 :11836      MOV WR[0] TO LS[L30]
U 2385. 6D61.15 :11837      BIS WR[2] TO LS[L30]
  
```

; LS = -1

; EXPECTED = 1

; CHECK WR.PLUS.LS

; LOOP ON ERROR IF ENABLED

; WR2 = 2

; LS = -1

; EXPECTED = 1

; CHECK WR.PLUS.LS

; LOOP ON ERROR IF ENABLED

; WR2 = 2

; LS = -1

; EXPECTED = 1

; CHECK WR.PLUS.LS

; LOOP ON ERROR IF ENABLED

; WR2 = 2

; LS = -1

; EXPECTED = 1

; CHECK WR.PLUS.LS

; LOOP ON ERROR IF ENABLED

; WR2 = AAAAAAAB

; LS = 55555555

; EXPECTED = FFFFFFFF

; CHECK WR.OR.LS

; LOOP ON ERROR IF ENABLED

; WR2 = AAAAAAAB

; LS = 55555555

ZZ
:
:

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

XXXXXXXXXXXX XXXXXXXXXXXX

```

U 23B8. 3698.95 :11893      MOV LS[ALTER.ODD] TO WR[1]
U 23B9. 4518.95 :11894      BIC LS[T12] TO WR[1]      ;EXPECTED = 0000AAAA
U 23BA. 0869.3C :11895      JSR [CHECK.RESULT]      ;CHECK WR.AND.LS
U 23BB. 8A3B.34 :11896      JMP [LOOP.TF.6.1BA]      ;LOOP ON ERROR IF ENABLED
U 23BC. 8A70.1C :11897      JSR [INC.OTHER]
                                LOOP.TF.6.1BB:
U 23BD. 3699.15 :11899      MOV LS[ALTER.ODD] TO WR[2]      ;WR2 = AAAAAAAA
U 23BE. 3628.15 :11900      MOV LS[FFFF] TO WR[0]
U 23BF. 3EE6.15 :11901      MOV WR[0] TO LS[L73]      ;LS = 0000FFFF
U 23C0. EEE7.15 :11902      AND WR[2] TO LS[L73]
U 23C1. B6E6.15 :11903      MOV LS[L73] TO WR[0]
U 23C2. 3698.95 :11904      MOV LS[ALTER.ODD] TO WR[1]
U 23C3. 4518.95 :11905      BIC LS[T12] TO WR[1]      ;EXPECTED = 0000AAAA
U 23C4. 0869.3C :11906      JSR [CHECK.RESULT]      ;CHECK WR.AND.LS
U 23C5. 0A3B.D4 :11907      JMP [LOOP.TF.6.1BB]      ;LOOP ON ERROR IF ENABLED
U 23C6. 8A70.1C :11908      JSR [INC.OTHER]
                                LOOP.TF.6.1BC:
U 23C7. 3699.15 :11910      MOV LS[ALTER.ODD] TO WR[2]      ;WR2 = AAAAAAAA
U 23C8. 3628.15 :11911      MOV LS[FFFF] TO WR[0]
U 23C9. 3E10.15 :11912      MOV WR[0] TO LS[T8]      ;LS = 0000FFFF
U 23CA. 6F11.15 :11913      XOR WR[2] TO LS[T8]
U 23CB. B610.15 :11914      MOV LS[T8] TO WR[0]
U 23CC. 3722.95 :11915      MOV LS[ALTER.MIX] TO WR[1]      ;EXPECTED = AAAA5555
U 23CD. 0869.3C :11916      JSR [CHECK.RESULT]      ;CHECK WR.XOR.LS
U 23CE. 8A3C.74 :11917      JMP [LOOP.TF.6.1BC]      ;LOOP ON ERROR IF ENABLED
U 23CF. 8A70.1C :11918      JSR [INC.OTHER]
                                LOOP.TF.6.1BD:
U 23D0. 3699.15 :11920      MOV LS[ALTER.ODD] TO WR[2]      ;WR2 = AAAAAAAA
U 23D1. 3628.15 :11921      MOV LS[FFFF] TO WR[0]
U 23D2. BE60.15 :11922      MOV WR[0] TO LS[L30]      ;LS = 0000FFFF
U 23D3. EF61.15 :11923      XOR WR[2] TO LS[L30]
U 23D4. 3660.15 :11924      MOV LS[L30] TO WR[0]
U 23D5. 3722.95 :11925      MOV LS[ALTER.MIX] TO WR[1]      ;EXPECTED = AAAA5555
U 23D6. 0869.3C :11926      JSR [CHECK.RESULT]      ;CHECK WR.XOR.LS
U 23D7. 8A3D.04 :11927      JMP [LOOP.TF.6.1BD]      ;LOOP ON ERROR IF ENABLED
U 23D8. 8A70.1C :11928      JSR [INC.OTHER]
                                LOOP.TF.6.1BE:
U 23D9. 3699.15 :11930      MOV LS[ALTER.ODD] TO WR[2]      ;WR2 = AAAAAAAA
U 23DA. 3628.15 :11931      MOV LS[FFFF] TO WR[0]
U 23DB. BEA6.15 :11932      MOV WR[0] TO LS[L53]      ;LS = 0000FFFF
U 23DC. EFA7.15 :11933      XOR WR[2] TO LS[L53]
U 23DD. 36A6.15 :11934      MOV LS[L53] TO WR[0]
U 23DE. 3722.95 :11935      MOV LS[ALTER.MIX] TO WR[1]      ;EXPECTED = AAAA5555
U 23DF. 0869.3C :11936      JSR [CHECK.RESULT]      ;CHECK WR.XOR.LS
U 23E0. 8A3D.94 :11937      JMP [LOOP.TF.6.1BE]      ;LOOP ON ERROR IF ENABLED
U 23E1. 8A70.1C :11938      JSR [INC.OTHER]
                                LOOP.TF.6.1BF:
U 23E2. 3699.15 :11940      MOV LS[ALTER.ODD] TO WR[2]      ;WR2 = AAAAAAAA
U 23E3. 3628.15 :11941      MOV LS[FFFF] TO WR[0]
U 23E4. 3EE6.15 :11942      MOV WR[0] TO LS[L73]      ;LS = 0000FFFF
U 23E5. 6FE7.15 :11943      XOR WR[2] TO LS[L73]
U 23E6. B6E6.15 :11944      MOV LS[L73] TO WR[0]
U 23E7. 3722.95 :11945      MOV LS[ALTER.MIX] TO WR[1]      ;EXPECTED = AAAA5555
U 23E8. 0869.3C :11946      JSR [CHECK.RESULT]      ;CHECK WR.XOR.LS
U 23E9. 0A3E.24 :11947      JMP [LOOP.TF.6.1BF]      ;LOOP ON ERROR IF ENABLED
  
```

```

U 23EA, 8A70,1C :11948 JSR [INC.OTHER]
:11949 LOOP.TF.6.1C0:
U 23EB, D842,15 :11950 MOV LS[#2] TO Q ; Q = 2
U 23EC, B69E,15 :11951 MOV LS[ONES] TO WR[0]
U 23ED, 3E10,15 :11952 MOV WR[0] TO LS[T8] ;LS = -1
U 23EE, 7010,15 :11953 ADD Q TO LS[T8]
U 23EF, B610,15 :11954 MOV LS[T8] TO WR[0]
U 23FO, 3640,95 :11955 MOV LS[#1] TO WR[1] ;EXPECTED = 1
U 23F1, 0869,3C :11956 JSR [CHECK.RESULT] ;CHECK Q.PLUS.LS
U 23F2, 0A3E,B4 :11957 JMP [LOOP.TF.6.1C0] ;LOOP ON ERROR IF ENABLED
U 23F3, 8A70,1C :11958 JSR [INC.OTHER]
:11959 LOOP.TF.6.1C1:
U 23F4, D842,15 :11960 MOV LS[#2] TO Q ; Q = 2
U 23F5, B69E,15 :11961 MOV LS[ONES] TO WR[0]
U 23F6, BE60,15 :11962 MOV WR[0] TO LS[L30] ;LS = -1
U 23F7, F060,15 :11963 ADD Q TO LS[L30]
U 23F8, 3660,15 :11964 MOV LS[L30] TO WR[0]
U 23F9, 3640,95 :11965 MOV LS[#1] TO WR[1] ;EXPECTED = 1
U 23FA, 0869,3C :11966 JSR [CHECK.RESULT] ;CHECK Q.PLUS.LS
U 23FB, 8A3F,44 :11967 JMP [LOOP.TF.6.1C1] ;LOOP ON ERROR IF ENABLED
U 23FC, 8A70,1C :11968 JSR [INC.OTHER]
:11969 LOOP.TF.6.1C2:
U 23FD, D842,15 :11970 MOV LS[#2] TO Q ; Q = 2
U 23FE, B69E,15 :11971 MOV LS[ONES] TO WR[0]
U 23FF, BEA6,15 :11972 MOV WR[0] TO LS[L53] ;LS = -1
U 2400, F0A6,15 :11973 ADD Q TO LS[L53]
U 2401, 36A6,15 :11974 MOV LS[L53] TO WR[0]
U 2402, 3640,95 :11975 MOV LS[#1] TO WR[1] ;EXPECTED = 1
U 2403, 0869,3C :11976 JSR [CHECK.RESULT] ;CHECK Q.PLUS.LS
U 2404, 8A3F,D4 :11977 JMP [LOOP.TF.6.1C2] ;LOOP ON ERROR IF ENABLED
U 2405, 8A70,1C :11978 JSR [INC.OTHER]
:11979 LOOP.TF.6.1C3:
U 2406, D842,15 :11980 MOV LS[#2] TO Q ; Q = 2
U 2407, B69E,15 :11981 MOV LS[ONES] TO WR[0]
U 2408, 3EE6,15 :11982 MOV WR[0] TO LS[L73] ;LS = -1
U 2409, 70E6,15 :11983 ADD Q TO LS[L73]
U 240A, B6E6,15 :11984 MOV LS[L73] TO WR[0]
U 240B, 3640,95 :11985 MOV LS[#1] TO WR[1] ;EXPECTED = 1
U 240C, 0869,3C :11986 JSR [CHECK.RESULT] ;CHECK Q.PLUS.LS
U 240D, 8A40,64 :11987 JMP [LOOP.TF.6.1C3] ;LOOP ON ERROR IF ENABLED
U 240E, 8A70,1C :11988 JSR [INC.OTHER]
U 240F, 0A41,A4 :11989 JMP [LOOP.TF.6.1C4]
:11990 241A:
:11991 LOOP.TF.6.1C4:
U 241A, 3642,15 :11992 MOV LS[#2] TO WR[0]
U 241B, 3E10,15 :11993 MOV WR[0] TO LS[T8] ;LS = 2
U 241C, 5840,15 :11994 MOV LS[#1] TO Q ; Q = 1
U 241D, F110,15 :11995 SUB LS[T8] FROM Q TO LS[T8]
U 241E, B610,15 :11996 MOV LS[T8] TO WR[0]
U 241F, 369E,95 :11997 MOV LS[ONES] TO WR[1] ;EXPECTED = -1
U 2420, 0869,3C :11998 JSR [CHECK.RESULT] ;CHECK LS.FROM.Q.LS
U 2421, 0A41,A4 :11999 JMP [LOOP.TF.6.1C4] ;LOOP ON ERROR IF ENABLED
U 2422, 8A70,1C :12000 JSR [INC.OTHER]
:12001 LOOP.TF.6.1C5:
U 2423, 3642,15 :12002 MOV LS[#2] TO WR[0]
  
```

```

U 2424, BE60,15 :12003      MOV WR[0] TO LS[L30]      ;LS = 2
U 2425, 5840,15 :12004      MOV LS[#1] TO Q      ; Q = 1
U 2426, 7160,15 :12005      SUB LS[L30] FROM Q TO LS[L30]
U 2427, 3660,15 :12006      MOV LS[L30] TO WR[0]
U 2428, 369E,95 :12007      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 2429, 0869,3C :12008      JSR [CHECK.RESULT]      ;CHECK LS.FROM.Q.LS
U 242A, 0A42,34 :12009      JMP [LOOP.TF.6.1C5]      ;LOOP ON ERROR IF ENABLED
U 242B, 8A70,1C :12010      JSR [INC.OTHER]
      :12011
      LOOP.TF.6.1C6:
U 242C, 3642,15 :12012      MOV LS[#2] TO WR[0]
U 242D, BEA6,15 :12013      MOV WR[0] TO LS[L53]      ;LS = 2
U 242E, 5840,15 :12014      MOV LS[#1] TO Q      ; Q = 1
U 242F, 71A6,15 :12015      SUB LS[L53] FROM Q TO LS[L53]
U 2430, 36A6,15 :12016      MOV LS[L53] TO WR[0]
U 2431, 369E,95 :12017      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 2432, 0869,3C :12018      JSR [CHECK.RESULT]      ;CHECK LS.FROM.Q.LS
U 2433, 0A42,C4 :12019      JMP [LOOP.TF.6.1C6]      ;LOOP ON ERROR IF ENABLED
U 2434, 8A70,1C :12020      JSR [INC.OTHER]
      :12021
      LOOP.TF.6.1C7:
U 2435, 3642,15 :12022      MOV LS[#2] TO WR[0]
U 2436, 3EE6,15 :12023      MOV WR[0] TO LS[L73]      ;LS = 2
U 2437, 5840,15 :12024      MOV LS[#1] TO Q      ; Q = 1
U 2438, F1E6,15 :12025      SUB LS[L73] FROM Q TO LS[L73]
U 2439, B6E6,15 :12026      MOV LS[L73] TO WR[0]
U 243A, 369E,95 :12027      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 243B, 0869,3C :12028      JSR [CHECK.RESULT]      ;CHECK LS.FROM.Q.LS
U 243C, 8A43,54 :12029      JMP [LOOP.TF.6.1C7]      ;LOOP ON ERROR IF ENABLED
U 243D, 8A70,1C :12030      JSR [INC.OTHER]
      :12031
      LOOP.TF.6.1C8:
U 243E, D842,15 :12032      MOV LS[#2] TO Q      ; Q = 2
U 243F, B640,15 :12033      MOV LS[#1] TO WR[0]
U 2440, 3E10,15 :12034      MOV WR[0] TO LS[T8]      ;LS = 1
U 2441, F210,15 :12035      SUB Q FROM LS[T8]
U 2442, B610,15 :12036      MOV LS[T8] TO WR[0]
U 2443, 369E,95 :12037      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 2444, 0869,3C :12038      JSR [CHECK.RESULT]      ;CHECK Q.FROM.LS
U 2445, 0A43,E4 :12039      JMP [LOOP.TF.6.1C8]      ;LOOP ON ERROR IF ENABLED
U 2446, 8A70,1C :12040      JSR [INC.OTHER]
      :12041
      LOOP.TF.6.1C9:
U 2447, D842,15 :12042      MOV LS[#2] TO Q      ; Q = 2
U 2448, B640,15 :12043      MOV LS[#1] TO WR[0]
U 2449, BE60,15 :12044      MOV WR[0] TO LS[L30]      ;LS = 1
U 244A, 7260,15 :12045      SUB Q FROM LS[L30]
U 244B, 3660,15 :12046      MOV LS[L30] TO WR[0]
U 244C, 369E,95 :12047      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 244D, 0869,3C :12048      JSR [CHECK.RESULT]      ;CHECK Q.FROM.LS
U 244E, 8A44,74 :12049      JMP [LOOP.TF.6.1C9]      ;LOOP ON ERROR IF ENABLED
U 244F, 8A70,1C :12050      JSR [INC.OTHER]
U 2450, 0A45,14 :12051      JMP [LOOP.TF.6.1CA]
      :12052
      LOOP.TF.6.1CA:
U 2451, D842,15 :12053      MOV LS[#2] TO Q      ; Q = 2
U 2452, B640,15 :12054      MOV LS[#1] TO WR[0]
U 2453, BEA6,15 :12055      MOV WR[0] TO LS[L53]      ;LS = 1
U 2454, 72A6,15 :12056      SUB Q FROM LS[L53]
U 2455, 36A6,15 :12057      MOV LS[L53] TO WR[0]
  
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040 1

```

U 2488, 0869,3C :12113 JSR [CHECK.RESULT] ;CHECK Q.OR.LS
U 2489, 8A48,14 :12114 JMP [LOOP.TF.6.1CF] ;LOOP ON ERROR IF ENABLED
U 248A, 8A70,1C :12115 JSR [INC.OTHER]
:12116 LOOP.TF.6.1D0:
U 248B, 5898,15 :12117 MOV LS[ALTER.ODD] TO Q ; Q = AAAAAAAA
U 248C, 3628,15 :12118 MOV LS[FFFF] TO WR[0]
U 248D, 3E10,15 :12119 MOV WR[0] TO LS[T8] ;LS = 0000FFFF
U 248E, F410,15 :12120 AND Q TO LS[T8]
U 248F, 8610,15 :12121 MOV LS[T8] TO WR[0]
U 2490, 3698,95 :12122 MOV LS[ALTER.ODD] TO WR[1]
U 2491, 4518,95 :12123 BIC LS[T12] TO WR[1] ;EXPECTED = 0000AAAA
U 2492, 0869,3C :12124 JSR [CHECK.RESULT] ;CHECK Q.AND.LS
U 2493, 8A48,B4 :12125 JMP [LOOP.TF.6.1D0] ;LOOP ON ERROR IF ENABLED
U 2494, 8A70,1C :12126 JSR [INC.OTHER]
:12127 LOOP.TF.6.1D1:
U 2495, 5898,15 :12128 MOV LS[ALTER.ODD] TO Q ; Q = AAAAAAAA
U 2496, 3628,15 :12129 MOV LS[FFFF] TO WR[0]
U 2497, BE60,15 :12130 MOV WR[0] TO LS[L30] ;LS = 0000FFFF
U 2498, 7460,15 :12131 AND Q TO LS[L30]
U 2499, 3660,15 :12132 MOV LS[L30] TO WR[0]
U 249A, 3698,95 :12133 MOV LS[ALTER.ODD] TO WR[1]
U 249B, 4518,95 :12134 BIC LS[T12] TO WR[1] ;EXPECTED = 0000AAAA
U 249C, 0869,3C :12135 JSR [CHECK.RESULT] ;CHECK Q.AND.LS
U 249D, 8A49,54 :12136 JMP [LOOP.TF.6.1D1] ;LOOP ON ERROR IF ENABLED
U 249E, 8A70,1C :12137 JSR [INC.OTHER]
:12138 LOOP.TF.6.1D2:
U 249F, 5898,15 :12139 MOV LS[ALTER.ODD] TO Q ; Q = AAAAAAAA
U 24A0, 3628,15 :12140 MOV LS[FFFF] TO WR[0]
U 24A1, BEA6,15 :12141 MOV WR[0] TO LS[L53] ;LS = 0000FFFF
U 24A2, 74A6,15 :12142 AND Q TO LS[L53]
U 24A3, 36A6,15 :12143 MOV LS[L53] TO WR[0]
U 24A4, 3698,95 :12144 MOV LS[ALTER.ODD] TO WR[1]
U 24A5, 4518,95 :12145 BIC LS[T12] TO WR[1] ;EXPECTED = 0000AAAA
U 24A6, 0869,3C :12146 JSR [CHECK.RESULT] ;CHECK Q.AND.LS
U 24A7, 8A49,F4 :12147 JMP [LOOP.TF.6.1D2] ;LOOP ON ERROR IF ENABLED
U 24A8, 8A70,1C :12148 JSR [INC.OTHER]
:12149 LOOP.TF.6.1D3:
U 24A9, 5898,15 :12150 MOV LS[ALTER.ODD] TO Q ; Q = AAAAAAAA
U 24AA, 3628,15 :12151 MOV LS[FFFF] TO WR[0]
U 24AB, 3EE6,15 :12152 MOV WR[0] TO LS[L73] ;LS = 0000FFFF
U 24AC, F4E6,15 :12153 AND Q TO LS[L73]
U 24AD, 86E6,15 :12154 MOV LS[L73] TO WR[0]
U 24AE, 3698,95 :12155 MOV LS[ALTER.ODD] TO WR[1]
U 24AF, 4518,95 :12156 BIC LS[T12] TO WR[1] ;EXPECTED = 0000AAAA
U 24B0, 0869,3C :12157 JSR [CHECK.RESULT] ;CHECK Q.AND.LS
U 24B1, 8A4A,94 :12158 JMP [LOOP.TF.6.1D3] ;LOOP ON ERROR IF ENABLED
U 24B2, 8A70,1C :12159 JSR [INC.OTHER]
:12160 LOOP.TF.6.1D4:
U 24B3, 5898,15 :12161 MOV LS[ALTER.ODD] TO Q ; Q = AAAAAAAA
U 24B4, 3628,15 :12162 MOV LS[FFFF] TO WR[0]
U 24B5, 3E10,15 :12163 MOV WR[0] TO LS[T8] ;LS = 0000FFFF
U 24B6, 7510,15 :12164 XOR Q TO LS[T8]
U 24B7, 8610,15 :12165 MOV LS[T8] TO WR[0]
U 24B8, 3722,95 :12166 MOV LS[ALTER.MIX] TO WR[1] ;EXPECTED = AAAA5555
U 24B9, 0869,3C :12167 JSR [CHECK.RESULT] ;CHECK Q.XOR.LS

```

```

U 24BA, 0A4B,34 :12168      JMP [LOOP.TF.6.1D4]          ;LOOP ON ERROR IF ENABLED
U 24BB, 8A70,1C :12169      JSR [INC.OTHER]
                                :12170
LOOP.TF.6.1D5:
U 24BC, 5898,15 :12171      MOV LS[ALTER.ODD] TO Q          ; Q = AAAAAAAA
U 24BD, 3628,15 :12172      MOV LS[FFFF] TO WR[0]
U 24BE, BE60,15 :12173      MOV WR[0] TO LS[L30]             ;LS = 0000FFFF
U 24BF, F560,15 :12174      XOR Q TO LS[L30]
U 24C0, 3660,15 :12175      MOV LS[L30] TO WR[0]
U 24C1, 3722,95 :12176      MOV LS[ALTER.MIX] TO WR[1]        ;EXPECTED = AAAA5555
U 24C2, 0869,3C :12177      JSR [CHECK.RESULT]               ;CHECK Q.XOR.LS
U 24C3, 0A4B,C4 :12178      JMP [LOOP.TF.6.1D5]              ;LOOP ON ERROR IF ENABLED
U 24C4, 8A70,1C :12179      JSR [INC.OTHER]
                                :12180
LOOP.TF.6.1D6:
U 24C5, 5898,15 :12181      MOV LS[ALTER.ODD] TO Q          ; Q = AAAAAAAA
U 24C6, 3628,15 :12182      MOV LS[FFFF] TO WR[0]
U 24C7, BEA6,15 :12183      MOV WR[0] TO LS[L53]             ;LS = 0000FFFF
U 24C8, F5A6,15 :12184      XOR Q TO LS[L53]
U 24C9, 36A6,15 :12185      MOV LS[L53] TO WR[0]
U 24CA, 3722,95 :12186      MOV LS[ALTER.MIX] TO WR[1]        ;EXPECTED = AAAA5555
U 24CB, 0869,3C :12187      JSR [CHECK.RESULT]               ;CHECK Q.XOR.LS
U 24CC, 8A4C,54 :12188      JMP [LOOP.TF.6.1D6]              ;LOOP ON ERROR IF ENABLED
U 24CD, 8A70,1C :12189      JSR [INC.OTHER]
                                :12190
LOOP.TF.6.1D7:
U 24CE, 5898,15 :12191      MOV LS[ALTER.ODD] TO Q          ; Q = AAAAAAAA
U 24CF, 3628,15 :12192      MOV LS[FFFF] TO WR[0]
U 24D0, 3EE6,15 :12193      MOV WR[0] TO LS[L73]             ;LS = 0000FFFF
U 24D1, 75E6,15 :12194      XOR Q TO LS[L73]
U 24D2, B6E6,15 :12195      MOV LS[L73] TO WR[0]
U 24D3, 3722,95 :12196      MOV LS[ALTER.MIX] TO WR[1]        ;EXPECTED = AAAA5555
U 24D4, 0869,3C :12197      JSR [CHECK.RESULT]               ;CHECK Q.XOR.LS
U 24D5, 0A4C,E4 :12198      JMP [LOOP.TF.6.1D7]              ;LOOP ON ERROR IF ENABLED
U 24D6, 8A70,1C :12199      JSR [INC.OTHER]
                                :12200
LOOP.TF.6.1D8:
U 24D7, D89A,15 :12201      MOV LS[ALTER.EVEN] TO Q          ; Q = 55555555
U 24D8, 7610,15 :12202      MOV Q TO LS[T8]
U 24D9, B610,15 :12203      MOV LS[T8] TO WR[0]
U 24DA, B69A,95 :12204      MOV LS[ALTER.EVEN] TO WR[1]        ;EXPECTED = 55555555
U 24DB, 0869,3C :12205      JSR [CHECK.RESULT]               ;CHECK MOV.Q.LS
U 24DC, 8A4D,74 :12206      JMP [LOOP.TF.6.1D8]              ;LOOP ON ERROR IF ENABLED
U 24DD, 8A70,1C :12207      JSR [INC.OTHER]
                                :12208
LOOP.TF.6.1D9:
U 24DE, D89A,15 :12209      MOV LS[ALTER.EVEN] TO Q          ; Q = 55555555
U 24DF, F660,15 :12210      MOV Q TO LS[L30]
U 24E0, 3660,15 :12211      MOV LS[L30] TO WR[0]
U 24E1, B69A,95 :12212      MOV LS[ALTER.EVEN] TO WR[1]        ;EXPECTED = 55555555
U 24E2, 0869,3C :12213      JSR [CHECK.RESULT]               ;CHECK MOV.Q.LS
U 24E3, 8A4D,E4 :12214      JMP [LOOP.TF.6.1D9]              ;LOOP ON ERROR IF ENABLED
U 24E4, 8A70,1C :12215      JSR [INC.OTHER]
                                :12216
LOOP.TF.6.1DA:
U 24E5, D89A,15 :12217      MOV LS[ALTER.EVEN] TO Q          ; Q = 55555555
U 24E6, F6A6,15 :12218      MOV Q TO LS[L53]
U 24E7, 36A6,15 :12219      MOV LS[L53] TO WR[0]
U 24E8, B69A,95 :12220      MOV LS[ALTER.EVEN] TO WR[1]        ;EXPECTED = 55555555
U 24E9, 0869,3C :12221      JSR [CHECK.RESULT]               ;CHECK MOV.Q.LS
U 24EA, 0A4F,54 :12222      JMP [LOOP.TF.6.1DA]              ;LOOP ON ERROR IF ENABLED
    
```

```

U 24EB, 8A70,1C :12223 JSR [INC.OTHER]
U 24EC, D89A,15 :12224 LOOP.TF.6.1DB:
U 24ED, 76E6,15 :12225 MOV LS[ALTER.EVEN] TO Q ; Q = 55555555
U 24EE, B6E6,15 :12226 MOV Q TO LS[L73]
U 24EF, B69A,95 :12227 MOV LS[L73] TO WR[0]
U 24F0, 0869,3C :12228 MOV LS[ALTER.EVEN] TO WR[1] ; EXPECTED = 55555555
U 24F1, 0A4E,C4 :12229 JSR [CHECK.RESULT] ; CHECK MOV.Q.LS
U 24F2, 8A70,1C :12230 JMP [LOOP.TF.6.1DB] ; LOOP ON ERROR IF ENABLED
U 24F3, 5840,15 :12231 JSR [INC.OTHER]
U 24F4, F710,15 :12232 LOOP.TF.6.1DC:
U 24F5, B610,15 :12233 MOV LS[#1] TO Q ; Q = 1
U 24F6, 369E,95 :12234 MNEG Q TO LS[T8]
U 24F7, 0869,3C :12235 MOV LS[T8] TO WR[0]
U 24F8, 8A4F,34 :12236 MOV LS[ONES] TO WR[1] ; EXPECTED = -1
U 24F9, 8A70,1C :12237 JSR [CHECK.RESULT] ; CHECK Q.NEG.LS
U 24FA, 5840,15 :12238 JMP [LOOP.TF.6.1DC] ; LOOP ON ERROR IF ENABLED
U 24FB, 7760,15 :12239 JSR [INC.OTHER]
U 24FC, 3660,15 :12240 LOOP.TF.6.1DD:
U 24FD, 369E,95 :12241 MOV LS[#1] TO Q ; Q = 1
U 24FE, 0869,3C :12242 MNEG Q TO LS[L30]
U 24FF, 8A4F,A4 :12243 MOV LS[L30] TO WR[0]
U 2500, 8A70,1C :12244 MOV LS[ONES] TO WR[1] ; EXPECTED = -1
U 2501, 5840,15 :12245 JSR [CHECK.RESULT] ; CHECK Q.NEG.LS
U 2502, 77A6,15 :12246 JMP [LOOP.TF.6.1DD] ; LOOP ON ERROR IF ENABLED
U 2503, 36A6,15 :12247 JSR [INC.OTHER]
U 2504, 369E,95 :12248 LOOP.TF.6.1DE:
U 2505, 0869,3C :12249 MOV LS[#1] TO Q ; Q = 1
U 2506, 8A50,14 :12250 MNEG Q TO LS[L53]
U 2507, 8A70,1C :12251 MOV LS[L53] TO WR[0]
U 2508, 5840,15 :12252 MOV LS[ONES] TO WR[1] ; EXPECTED = -1
U 2509, F7E6,15 :12253 JSR [CHECK.RESULT] ; CHECK Q.NEG.LS
U 250A, B6E6,15 :12254 JMP [LOOP.TF.6.1DE] ; LOOP ON ERROR IF ENABLED
U 250B, 369E,95 :12255 JSR [INC.OTHER]
U 250C, 0869,3C :12256 LOOP.TF.6.1DF:
U 250D, 8A50,84 :12257 MOV LS[#1] TO Q ; Q = 1
U 250E, 8A70,1C :12258 MNEG Q TO LS[L73]
U 250F, B698,15 :12259 MOV LS[L73] TO WR[0]
U 2510, 7811,15 :12260 MOV LS[ONES] TO WR[1] ; EXPECTED = -1
U 2511, B610,15 :12261 JSR [CHECK.RESULT] ; CHECK Q.NEG.LS
U 2512, 3698,95 :12262 JMP [LOOP.TF.6.1DF] ; LOOP ON ERROR IF ENABLED
U 2513, 0869,3C :12263 JSR [INC.OTHER]
U 2514, 0A50,F4 :12264 LOOP.TF.6.1E0:
U 2515, 8A70,1C :12265 MOV LS[ALTER.EVEN] TO WR[2] ; WR2 = 55555555
U 2516, B698,15 :12266 MCOM WR[2] TO LS[T8]
U 2517, F861,15 :12267 MOV LS[T8] TO WR[0]
U 2518, 3660,15 :12268 MOV LS[ALTER.ODD] TO WR[1] ; EXPECTED = AAAAAAAA
U 2519, 3698,95 :12269 JSR [CHECK.RESULT] ; CHECK WR.COM.LS
U 251A, 0869,3C :12270 JMP [LOOP.TF.6.1E0] ; LOOP ON ERROR IF ENABLED
U 251B, 8A70,1C :12271 JSR [INC.OTHER]
U 251C, B698,15 :12272 LOOP.TF.6.1E1:
U 251D, F861,15 :12273 MOV LS[ALTER.EVEN] TO WR[2] ; WR2 = 55555555
U 251E, 3660,15 :12274 MCOM WR[2] TO LS[L30]
U 251F, 3698,95 :12275 MOV LS[L30] TO WR[0]
U 2520, 0869,3C :12276 MOV LS[ALTER.ODD] TO WR[1] ; EXPECTED = AAAAAAAA
U 2521, 8A70,1C :12277 JSR [CHECK.RESULT] ; CHECK WR.COM.LS

```



```

U 251B, 8A51,64 :12278      JMP [LOOP.TF.6.1E1]      ;LOOP ON ERROR IF ENABLED
U 251C, 8A70,1C :12279      JSR [INC.OTHER]
:12280
U 251D, B69B,15 :12281      LOOP.TF.6.1E2:      ;WR2 = 55555555
U 251E, F8A7,15 :12282      MOV LS[ALTER.EVEN] TO WR[2]
U 251F, 36A6,15 :12283      MCOM WR[2] TO LS[L53]
U 2520, 3698,95 :12284      MOV LS[L53] TO WR[0]
U 2521, 0869,3C :12285      MOV LS[ALTER.ODD] TO WR[1]      ;EXPECTED = AAAAAAAA
U 2522, 0A51,D4 :12286      JSR [CHECK.RESULT]      ;CHECK WR.COM.LS
U 2523, 8A70,1C :12287      JMP [LOOP.TF.6.1E2]      ;LOOP ON ERROR IF ENABLED
:12288
U 2524, B69B,15 :12289      LOOP.TF.6.1E3:      ;WR2 = 55555555
U 2525, 78E7,15 :12290      MOV LS[ALTER.EVEN] TO WR[2]
U 2526, B6E6,15 :12291      MCOM WR[2] TO LS[L73]
U 2527, 3698,95 :12292      MOV LS[L73] TO WR[0]
U 2528, 0869,3C :12293      MOV LS[ALTER.ODD] TO WR[1]      ;EXPECTED = AAAAAAAA
U 2529, 0A52,44 :12294      JSR [CHECK.RESULT]      ;CHECK WR.COM.LS
U 252A, 8A70,1C :12295      JMP [LOOP.TF.6.1E3]      ;LOOP ON ERROR IF ENABLED
:12296
U 252B, 3641,15 :12297      LOOP.TF.6.1E4:      ;WR2 = 1
U 252C, F911,15 :12298      MOV LS[#1] TO WR[2]
U 252D, B610,15 :12299      MNEG WR[2] TO LS[T8]
U 252E, 369E,95 :12300      MOV LS[T8] TO WR[0]
U 252F, 0869,3C :12301      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 2530, 0A52,B4 :12302      JSR [CHECK.RESULT]      ;CHECK WR.NEG.LS
U 2531, 8A70,1C :12303      JMP [LOOP.TF.6.1E4]      ;LOOP ON ERROR IF ENABLED
:12304
U 2532, 3641,15 :12305      LOOP.TF.6.1E5:      ;WR2 = 1
U 2533, 7961,15 :12306      MOV LS[#1] TO WR[2]
U 2534, 3660,15 :12307      MNEG WR[2] TO LS[L30]
U 2535, 369E,95 :12308      MOV LS[L30] TO WR[0]
U 2536, 0869,3C :12309      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 2537, 8A53,24 :12310      JSR [CHECK.RESULT]      ;CHECK WR.NEG.LS
U 2538, 8A70,1C :12311      JMP [LOOP.TF.6.1E5]      ;LOOP ON ERROR IF ENABLED
:12312
U 2539, 3641,15 :12313      LOOP.TF.6.1E6:      ;WR2 = 1
U 253A, 79A7,15 :12314      MOV LS[#1] TO WR[2]
U 253B, 36A6,15 :12315      MNEG WR[2] TO LS[L53]
U 253C, 369E,95 :12316      MOV LS[L53] TO WR[0]
U 253D, 0869,3C :12317      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 253E, 0A53,94 :12318      JSR [CHECK.RESULT]      ;CHECK WR.NEG.LS
U 253F, 8A70,1C :12319      JMP [LOOP.TF.6.1E6]      ;LOOP ON ERROR IF ENABLED
:12320
U 2540, 3641,15 :12321      LOOP.TF.6.1E7:      ;WR2 = 1
U 2541, F9E7,15 :12322      MOV LS[#1] TO WR[2]
U 2542, B6E6,15 :12323      MNEG WR[2] TO LS[L73]
U 2543, 369E,95 :12324      MOV LS[L73] TO WR[0]
U 2544, 0869,3C :12325      MOV LS[ONES] TO WR[1]      ;EXPECTED = -1
U 2545, 8A54,04 :12326      JSR [CHECK.RESULT]      ;CHECK WR.NEG.LS
U 2546, 8A70,1C :12327      JMP [LOOP.TF.6.1E7]      ;LOOP ON ERROR IF ENABLED
:12328
U 2547, D842,15 :12329      LOOP.TF.6.1E8:
U 2548, 369F,15 :12330      MOV LS[#2] TO 0      ; 0 = 2
U 2549, FA11,15 :12331      MOV LS[ONES] TO WR[2]      ;WR2 = -1
U 254A, B610,15 :12332      ADD 0 PLUS WR[2] TO LS[T8]
:12332
MOV LS[T8] TO WR[0]

```

```

U 254B, 3640,95 :12333      MOV LS[#1] TO WR[1]          ;EXPECTED = 1
U 254C, 0869,3C :12334      JSR [CHECK.RESULT]        ;CHECK Q.PLUS.WR.TO.LS
U 254D, 0A54,74 :12335      JMP [LOOP.TF.6.1E8]        ;LOOP ON ERROR IF ENABLED
U 254E, 8A70,1C :12336      JSR [INC.OTHER]
U 254F, D842,15 :12337      LOOP.TF.6.1E9:
U 2550, 369F,15 :12338      MOV LS[#2] TO Q              ; Q = 2
U 2551, 7A61,15 :12339      MOV LS[ONES] TO WR[2]        ;WR2 = -1
U 2552, 3660,15 :12340      ADD Q PLUS WR[2] TO LS[L30]
U 2553, 3640,95 :12341      MOV LS[L30] TO WR[0]
U 2554, 0869,3C :12342      MOV LS[#1] TO WR[1]          ;EXPECTED = 1
U 2555, 8A54,F4 :12343      JSR [CHECK.RESULT]        ;CHECK Q.PLUS.WR.TO.LS
U 2556, 8A70,1C :12344      JMP [LOOP.TF.6.1E9]        ;LOOP ON ERROR IF ENABLED
U 2557, D842,15 :12345      JSR [INC.OTHER]
U 2558, 369F,15 :12346      LOOP.TF.6.1EA:
U 2559, 7AA7,15 :12347      MOV LS[#2] TO Q              ; Q = 2
U 255A, 36A6,15 :12348      MOV LS[ONES] TO WR[2]        ;WR2 = -1
U 255B, 3640,95 :12349      ADD Q PLUS WR[2] TO LS[L53]
U 255C, 0869,3C :12350      MOV LS[L53] TO WR[0]
U 255D, 8A55,F4 :12351      MOV LS[#1] TO WR[1]          ;EXPECTED = 1
U 255E, 8A70,1C :12352      JSR [CHECK.RESULT]        ;CHECK Q.PLUS.WR.TO.LS
U 255F, D842,15 :12353      JMP [LOOP.TF.6.1EA]        ;LOOP ON ERROR IF ENABLED
U 2560, 369F,15 :12354      JSR [INC.OTHER]
U 2561, FAE7,15 :12355      LOOP.TF.6.1EB:
U 2562, B6E6,15 :12356      MOV LS[#2] TO Q              ; Q = 2
U 2563, 3640,95 :12357      MOV LS[ONES] TO WR[2]        ;WR2 = -1
U 2564, 0869,3C :12358      ADD Q PLUS WR[2] TO LS[L73]
U 2565, 0A55,F4 :12359      MOV LS[L73] TO WR[0]
U 2566, 8A70,1C :12360      MOV LS[#1] TO WR[1]          ;EXPECTED = 1
U 2567, D842,15 :12361      JSR [CHECK.RESULT]        ;CHECK Q.PLUS.WR.TO.LS
U 2568, 3641,15 :12362      JMP [LOOP.TF.6.1EB]        ;LOOP ON ERROR IF ENABLED
U 2569, 7B11,15 :12363      JSR [INC.OTHER]
U 256A, B610,15 :12364      LOOP.TF.6.1EC:
U 256B, 369E,95 :12365      MOV LS[#2] TO Q              ; Q = 2
U 256C, 0869,3C :12366      MOV LS[#1] TO WR[2]        ;WR2 = 1
U 256D, 8A56,74 :12367      SUB Q FROM WR[2] TO LS[T8]
U 256E, 8A70,1C :12368      MOV LS[T8] TO WR[0]
U 256F, D842,15 :12369      MOV LS[ONES] TO WR[1]        ;EXPECTED = -1
U 2570, 3641,15 :12370      JSR [CHECK.RESULT]        ;CHECK Q.FROM.WR.TO.LS
U 2571, FB61,15 :12371      JMP [LOOP.TF.6.1EC]        ;LOOP ON ERROR IF ENABLED
U 2572, 3660,15 :12372      JSR [INC.OTHER]
U 2573, 369E,95 :12373      LOOP.TF.6.1ED:
U 2574, 0869,3C :12374      MOV LS[#2] TO Q              ; Q = 2
U 2575, 0A56,F4 :12375      MOV LS[#1] TO WR[2]        ;WR2 = 1
U 2576, 8A70,1C :12376      SUB Q FROM WR[2] TO LS[L30]
U 2577, D842,15 :12377      MOV LS[L30] TO WR[0]
U 2578, 3641,15 :12378      MOV LS[ONES] TO WR[1]        ;EXPECTED = -1
U 2579, FB61,15 :12379      JSR [CHECK.RESULT]        ;CHECK Q.FROM.WR.TO.LS
U 257A, 36A6,15 :12380      JMP [LOOP.TF.6.1ED]        ;LOOP ON ERROR IF ENABLED
U 257B, 369E,95 :12381      JSR [INC.OTHER]
U 257C, D842,15 :12382      LOOP.TF.6.1EE:
U 257D, 3641,15 :12383      MOV LS[#2] TO Q              ; Q = 2
U 257E, FB61,15 :12384      MOV LS[#1] TO WR[2]        ;WR2 = 1
U 257F, 36A6,15 :12385      SUB Q FROM WR[2] TO LS[L53]
U 2580, 369E,95 :12386      MOV LS[L53] TO WR[0]
U 2581, 369E,95 :12387      MOV LS[ONES] TO WR[1]        ;EXPECTED = -1
  
```

```

U 257C, 0869,3C :12388 JSR [CHECK.RESULT] ;CHECK Q.FROM.WR.TO.LS
U 257D, 0A57,74 :12389 JMP [LOOP.TF.6.1EF] ;LOOP ON ERROR IF ENABLED
U 257E, 8A70,1C :12390 JSR [INC.OTHER]
:12391
LOOP.TF.6.1EF:
U 257F, 0842,15 :12392 MOV LS[#2] TO Q ; Q = 2
U 2580, 3641,15 :12393 MOV LS[#1] TO WR[2] ;WR2 = 1
U 2581, 7BE7,15 :12394 SUB Q FROM WR[2] TO LS[L73]
U 2582, B6E6,15 :12395 MOV LS[L73] TO WR[0]
U 2583, 369E,95 :12396 MOV LS[ONES] TO WR[1] ;EXPECTED = -1
U 2584, 0869,3C :12397 JSR [CHECK.RESULT] ;CHECK Q.FROM.WR.TO.LS
U 2585, 8A57,F4 :12398 JMP [LOOP.TF.6.1EF] ;LOOP ON ERROR IF ENABLED
U 2586, 8A70,1C :12399 JSR [INC.OTHER]
:12400
LOOP.TF.6.1F0:
U 2587, 369C,15 :12401 MOV LS[ZERO] TO WR[0]
U 2588, 3E10,15 :12402 MOV WR[0] TO LS[T8] ;LS = 0
U 2589, 7C10,15 :12403 DEC LS[T8]
U 258A, B610,15 :12404 MOV LS[T8] TO WR[0]
U 258B, 369E,95 :12405 MOV LS[ONES] TO WR[1] ;EXPECTED = -1
U 258C, 0869,3C :12406 JSR [CHECK.RESULT] ;CHECK DEC.LS
U 258D, 0A58,74 :12407 JMP [LOOP.TF.6.1F0] ;LOOP ON ERROR IF ENABLED
U 258E, 8A70,1C :12408 JSR [INC.OTHER]
:12409
LOOP.TF.6.1F1:
U 258F, 369C,15 :12410 MOV LS[ZERO] TO WR[0]
U 2590, BE60,15 :12411 MOV WR[0] TO LS[L30] ;LS = 0
U 2591, FC60,15 :12412 DEC LS[L30]
U 2592, 3660,15 :12413 MOV LS[L30] TO WR[0]
U 2593, 369E,95 :12414 MOV LS[ONES] TO WR[1] ;EXPECTED = -1
U 2594, 0869,3C :12415 JSR [CHECK.RESULT] ;CHECK DEC.LS
U 2595, 8A58,F4 :12416 JMP [LOOP.TF.6.1F1] ;LOOP ON ERROR IF ENABLED
U 2596, 8A70,1C :12417 JSR [INC.OTHER]
:12418
LOOP.TF.6.1F2:
U 2597, 369C,15 :12419 MOV LS[ZERO] TO WR[0]
U 2598, BEA6,15 :12420 MOV WR[0] TO LS[L53] ;LS = 0
U 2599, FCA6,15 :12421 DEC LS[L53]
U 259A, 36A6,15 :12422 MOV LS[L53] TO WR[0]
U 259B, 369E,95 :12423 MOV LS[ONES] TO WR[1] ;EXPECTED = -1
U 259C, 0869,3C :12424 JSR [CHECK.RESULT] ;CHECK DEC.LS
U 259D, 8A59,74 :12425 JMP [LOOP.TF.6.1F2] ;LOOP ON ERROR IF ENABLED
U 259E, 8A70,1C :12426 JSR [INC.OTHER]
:12427
LOOP.TF.6.1F3:
U 259F, 369C,15 :12428 MOV LS[ZERO] TO WR[0]
U 25A0, 3EE6,15 :12429 MOV WR[0] TO LS[L73] ;LS = 0
U 25A1, 7CE6,15 :12430 DEC LS[L73]
U 25A2, B6E6,15 :12431 MOV LS[L73] TO WR[0]
U 25A3, 369E,95 :12432 MOV LS[ONES] TO WR[1] ;EXPECTED = -1
U 25A4, 0869,3C :12433 JSR [CHECK.RESULT] ;CHECK DEC.LS
U 25A5, 0A59,F4 :12434 JMP [LOOP.TF.6.1F3] ;LOOP ON ERROR IF ENABLED
U 25A6, 8A70,1C :12435 JSR [INC.OTHER]
:12436
LOOP.TF.6.1F4:
U 25A7, 369A,15 :12437 MOV LS[ALTER.EVEN] TO WR[0]
U 25A8, 3E10,15 :12438 MOV WR[0] TO LS[T8] ;LS = 55555555
U 25A9, FD10,15 :12439 COM LS[T8]
U 25AA, B610,15 :12440 MOV LS[T8] TO WR[0]
U 25AB, 3698,95 :12441 MOV LS[ALTER.ODD] TO WR[1] ;EXPECTED = AAAAAAAA
U 25AC, 0869,3C :12442 JSR [CHECK.RESULT] ;CHECK COM.LS
  
```

```

U 25AD, 8A5A,74 :12443      JMP [LOOP.TF.6.1F4]          ;LOOP ON ERROR IF ENABLED
U 25AE, 8A70,1C :12444      JSR [INC.OTHER]
U 25AF, 369A,15 :12445      LOOP.TF.6.1F5:
U 25B0, BE60,15 :12446      MOV LS[ALTER.EVEN] TO WR[0]
U 25B1, 7D60,15 :12447      MOV WR[0] TO LS[L30]          ;LS = 55555555
U 25B2, 3660,15 :12448      COM LS[L30]
U 25B3, 3698,95 :12449      MOV LS[L30] TO WR[0]
U 25B4, 0869,3C :12450      MOV LS[ALTER.ODD] TO WR[1]      ;EXPECTED = AAAAAAAA
U 25B5, 0A5A,F4 :12451      JSR [CHECK.RESULT]             ;CHECK COM.LS
U 25B6, 8A70,1C :12452      JMP [LOOP.TF.6.1F5]          ;LOOP ON ERROR IF ENABLED
U 25B7, 369A,15 :12453      JSR [INC.OTHER]
U 25B8, BEA6,15 :12454      LOOP.TF.6.1F6:
U 25B9, 7DA6,15 :12455      MOV LS[ALTER.EVEN] TO WR[0]
U 25BA, 36A6,15 :12456      MOV WR[0] TO LS[L53]          ;LS = 55555555
U 25BB, 3698,95 :12457      COM LS[L53]
U 25BC, 0869,3C :12458      MOV LS[L53] TO WR[0]
U 25BD, 0A5B,74 :12459      MOV LS[ALTER.ODD] TO WR[1]      ;EXPECTED = AAAAAAAA
U 25BE, 8A70,1C :12460      JSR [CHECK.RESULT]             ;CHECK COM.LS
U 25BF, 369A,15 :12461      JMP [LOOP.TF.6.1F6]          ;LOOP ON ERROR IF ENABLED
U 25C0, 3EE6,15 :12462      JSR [INC.OTHER]
U 25C1, FDE6,15 :12463      LOOP.TF.6.1F7:
U 25C2, B6E6,15 :12464      MOV LS[ALTER.EVEN] TO WR[0]
U 25C3, 3698,95 :12465      MOV WR[0] TO LS[L73]          ;LS = 55555555
U 25C4, 0869,3C :12466      COM LS[L73]
U 25C5, 8A5B,F4 :12467      MOV LS[L73] TO WR[0]
U 25C6, 8A70,1C :12468      MOV LS[ALTER.ODD] TO WR[1]      ;EXPECTED = AAAAAAAA
U 25C7, B69E,15 :12469      JSR [CHECK.RESULT]             ;CHECK COM.LS
U 25C8, 3E10,15 :12470      JMP [LOOP.TF.6.1F7]          ;LOOP ON ERROR IF ENABLED
U 25C9, FE10,15 :12471      JSR [INC.OTHER]
U 25CA, B610,15 :12472      LOOP.TF.6.1F8:
U 25CB, 3640,95 :12473      MOV LS[ONES] TO WR[0]
U 25CC, 0869,3C :12474      MOV WR[0] TO LS[T8]          ;LS = -1
U 25CD, 8A5C,74 :12475      NEG LS[T8]
U 25CE, 8A70,1C :12476      MOV LS[T8] TO WR[0]
U 25CF, B69E,15 :12477      MOV LS[#1] TO WR[1]            ;EXPECTED = 1
U 25D0, BE60,15 :12478      JSR [CHECK.RESULT]             ;CHECK NEG.LS
U 25D1, 7E60,15 :12479      JMP [LOOP.TF.6.1F8]          ;LOOP ON ERROR IF ENABLED
U 25D2, 3660,15 :12480      JSR [INC.OTHER]
U 25D3, 3640,95 :12481      LOOP.TF.6.1F9:
U 25D4, 0869,3C :12482      MOV LS[ONES] TO WR[0]
U 25D5, 0A5C,F4 :12483      MOV WR[0] TO LS[L30]          ;LS = -1
U 25D6, 8A70,1C :12484      NEG LS[L30]
U 25D7, B69E,15 :12485      MOV LS[L30] TO WR[0]
U 25D8, BEA6,15 :12486      MOV LS[#1] TO WR[1]            ;EXPECTED = 1
U 25D9, 7EA6,15 :12487      JSR [CHECK.RESULT]             ;CHECK NEG.LS
U 25DA, 36A6,15 :12488      JMP [LOOP.TF.6.1F9]          ;LOOP ON ERROR IF ENABLED
U 25DB, 3640,95 :12489      JSR [INC.OTHER]
U 25DC, 0869,3C :12490      LOOP.TF.6.1FA:
U 25DD, 0A5D,74 :12491      MOV LS[ONES] TO WR[0]
U 25DE, BEA6,15 :12492      MOV WR[0] TO LS[L53]          ;LS = -1
U 25DF, 7EA6,15 :12493      NEG LS[L53]
U 25E0, 36A6,15 :12494      MOV LS[L53] TO WR[0]
U 25E1, 3640,95 :12495      MOV LS[#1] TO WR[1]            ;EXPECTED = 1
U 25E2, 0869,3C :12496      JSR [CHECK.RESULT]             ;CHECK NEG.LS
U 25E3, 0A5D,74 :12497      JMP [LOOP.TF.6.1FA]          ;LOOP ON ERROR IF ENABLED
  
```

```

U 25DE, 8A70,1C :12498 JSR [INC.OTHER]
:12499
U 25DF, B69E,15 :12500 LOOP.TF.6.1FB:
U 25E0, 3EE6,15 :12501 MOV LS[ONES] TO WR[0]
U 25E1, FEE6,15 :12502 MOV WR[0] TO LS[L73] ;LS = -1
U 25E2, B6E6,15 :12503 NEG LS[L73]
U 25E3, 3640,95 :12504 MOV LS[L73] TO WR[0]
U 25E4, 0869,3C :12505 MOV LS[#1] TO WR[1] ;EXPECTED = 1
U 25E5, 8A5D,F4 :12506 JSR [CHECK.RESULT] ;CHECK NEG.LS
U 25E6, 8A70,1C :12507 JMP [LOOP.TF.6.1FB] ;LOOP ON ERROR IF ENABLED
:12508 JSR [INC.OTHER]
U 25E7, 2F80,15 :12509 LOOP.TF.6.1FC:
U 25E8, 3E10,15 :12510 CLR WR[0]
U 25E9, 7F10,15 :12511 MOV WR[0] TO LS[T8] ;LS = 0
U 25EA, B610,15 :12512 INC LS[T8]
U 25EB, 3640,95 :12513 MOV LS[T8] TO WR[0]
U 25EC, 0869,3C :12514 MOV LS[#1] TO WR[1] ;EXPECTED = 1
U 25ED, 0A5E,74 :12515 JSR [CHECK.RESULT] ;CHECK INC.LS
U 25EE, 8A70,1C :12516 JMP [LOOP.TF.6.1FC] ;LOOP ON ERROR IF ENABLED
:12517 JSR [INC.OTHER]
U 25EF, 2F80,15 :12518 LOOP.TF.6.1FD:
U 25F0, BE60,15 :12519 CLR WR[0]
U 25F1, FF60,15 :12520 MOV WR[0] TO LS[L30] ;LS = 0
U 25F2, 3660,15 :12521 INC LS[L30]
U 25F3, 3640,95 :12522 MOV LS[L30] TO WR[0]
U 25F4, 0869,3C :12523 MOV LS[#1] TO WR[1] ;EXPECTED = 1
U 25F5, 8A5E,F4 :12524 JSR [CHECK.RESULT] ;CHECK INC.LS
U 25F6, 8A70,1C :12525 JMP [LOOP.TF.6.1FD] ;LOOP ON ERROR IF ENABLED
:12526 JSR [INC.OTHER]
U 25F7, 2F80,15 :12527 LOOP.TF.6.1FE:
U 25F8, BEA6,15 :12528 CLR WR[0]
U 25F9, FFA6,15 :12529 MOV WR[0] TO LS[L53] ;LS = 0
U 25FA, 36A6,15 :12530 INC LS[L53]
U 25FB, 3640,95 :12531 MOV LS[L53] TO WR[0]
U 25FC, 0869,3C :12532 MOV LS[#1] TO WR[1] ;EXPECTED = 1
U 25FD, 8A5F,74 :12533 JSR [CHECK.RESULT] ;CHECK INC.LS
U 25FE, 8A70,1C :12534 JMP [LOOP.TF.6.1FE] ;LOOP ON ERROR IF ENABLED
:12535 JSR [INC.OTHER]
U 25FF, 2F80,15 :12536 LOOP.TF.6.1FF:
U 2600, 3EE6,15 :12537 CLR WR[0]
U 2601, 7FE6,15 :12538 MOV WR[0] TO LS[L73] ;LS = 0
U 2602, B6E6,15 :12539 INC LS[L73]
U 2603, 3640,95 :12540 MOV LS[L73] TO WR[0]
U 2604, 0869,3C :12541 MOV LS[#1] TO WR[1] ;EXPECTED = 1
U 2605, 0A5F,F4 :12542 JSR [CHECK.RESULT] ;CHECK INC.LS
:12543 JMP [LOOP.TF.6.1FF] ;LOOP ON ERROR IF ENABLED
:12544
:12545
:12546
:12547
:12548
:12549
:12550
:12551
:12552

```

MOVE instruction tests

The MOVE Instructions will be tested in groups of eight, the same instruction
 being tested for eight different ranges of Local Store: 0-1F, 20-3F, 40-5F,
 60-7F, 80-9F, A0-BF, C0-DF, and E0-FF. The eight actual locations used will
 be:

LS 8 - T8
 LS 30 - #26
 LS 53 - #F

```

:12553 :      LS 73 - #3F
:12554 :      LS 90 - L90
:12555 :      LS B0 - LB0
:12556 :      LS D0 - LD0
:12557 : and    LS F0 - LF0
:12558 :
:12559 : NOTE: Not all the move instructions are tested in this module. All moves that
:12560 : require Memory to have been previously tested will be tested in the next
:12561 : module.
:12562 :
:12563 : ff.7:
U 2606, 8870.0C :12564 :      JSR [INC.EN]
U 2607, 3720.15 :12565 :      MOV LS[L90] TO WR[0]
U 2608, 3760.95 :12566 :      MOV LS[LB0] TO WR[1]
U 2609, 37A1.15 :12567 :      MOV LS[LD0] TO WR[2]
U 260A, 37E1.95 :12568 :      MOV LS[LF0] TO WR[3]
U 260B, BE18.15 :12569 :      MOV WR[0] TO LS[T12]
U 260C, BE1A.95 :12570 :      MOV WR[1] TO LS[T13]
U 260D, BE1D.15 :12571 :      MOV WR[2] TO LS[T14]
U 260E, BE1F.95 :12572 :      MOV WR[3] TO LS[T15]
U 260F, 364E.15 :12573 :      MOV LS[BIT7] TO WR[0]
U 2610, C74C.15 :12574 :      BIS LS[BIT6] TO WR[0]
U 2611, 4748.15 :12575 :      BIS LS[BIT4] TO WR[0]
U 2612, C746.15 :12576 :      BIS LS[BIT3] TO WR[0]
U 2613, BE88.15 :12577 :      MOV WR[0] TO LS[ADDRESS.DATA]
:12578 :
:12579 : LOOP.TF.7.D8:
U 2614, B698.15 :12580 :      MOV LS[ALTER.ODD] TO WR[0]
U 2615, 3E10.15 :12581 :      MOV WR[0] TO LS[T8]
U 2616, 2F80.15 :12582 :      CLR WR[0]
U 2617, B610.15 :12583 :      MOV LS[T8] TO WR[0]
U 2618, 3698.95 :12584 :      MOV LS[ALTER.ODD] TO WR[1]
U 2619, 0869.3C :12585 :      JSR [CHECK.RESULT]
U 261A, 0A61.44 :12586 :      JMP [LOOP.TF.7.D8]
U 261B, 8A70.1C :12587 :      JSR [INC.OTHER]
:12588 :
:12589 : LOOP.TF.7.D9:
U 261C, B698.15 :12590 :      MOV LS[ALTER.ODD] TO WR[0]
U 261D, BE60.15 :12591 :      MOV WR[0] TO LS[L30]
U 261E, 2F80.15 :12592 :      CLR WR[0]
U 261F, 3660.15 :12593 :      MOV LS[L30] TO WR[0]
U 2620, 3698.95 :12594 :      MOV LS[ALTER.ODD] TO WR[1]
U 2621, 0869.3C :12595 :      JSR [CHECK.RESULT]
U 2622, 8A61.C4 :12596 :      JMP [LOOP.TF.7.D9]
U 2623, 8A70.1C :12597 :      JSR [INC.OTHER]
:12598 :
:12599 : LOOP.TF.7.DA:
U 2624, B698.15 :12600 :      MOV LS[ALTER.ODD] TO WR[0]
U 2625, BEA6.15 :12601 :      MOV WR[0] TO LS[L53]
U 2626, 2F80.15 :12602 :      CLR WR[0]
U 2627, 36A6.15 :12603 :      MOV LS[L53] TO WR[0]
U 2628, 3698.95 :12604 :      MOV LS[ALTER.ODD] TO WR[1]
U 2629, 0869.3C :12605 :      JSR [CHECK.RESULT]
U 262A, 0A62.44 :12606 :      JMP [LOOP.TF.7.DA]
U 262B, 8A70.1C :12607 :      JSR [INC.OTHER]
:12608 :
:12609 : LOOP.TF.7.DB:
U 262C, B698.15 :12610 :      MOV LS[ALTER.ODD] TO WR[0]
U 262D, 3EE6.15 :12611 :      MOV WR[0] TO LS[L73]

```

```

: INCREMENT ERROR NUMBER TO 7
: SAVE LS 90, B0, D0 AND F0

```

```

: INIT 'OTHER' DATA FIELD TO D8

```

```

: PUT DATA IN LS

```

```

: EXPECTED = SAME
: CHECK MOV.LS.WR
: LOOP ON ERROR IF ENABLED

```

```

: PUT DATA IN LS

```

```

: EXPECTED = SAME
: CHECK MOV.LS.WR
: LOOP ON ERROR IF ENABLED

```

```

: PUT DATA IN LS

```

```

: EXPECTED = SAME
: CHECK MOV.LS.WR
: LOOP ON ERROR IF ENABLED

```

```

: PUT DATA IN LS

```

```

U 262E, 2F80.15 :12608 CLR WR[0]
U 262F, B6E6.15 :12609 MOV LS[L73] TO WR[0]
U 2630, 3698.95 :12610 MOV LS[ALTER.ODD] TO WR[1] ;EXPECTED = SAME
U 2631, 0869.3C :12611 JSR [CHECK.RESULT] ;CHECK MOV.LS.WR
U 2632, 8A62.C4 :12612 JMP [LOOP.TF.7.DB] ;LOOP ON ERROR IF ENABLED
U 2633, 8A70.1C :12613 JSR [INC.OTHER]
:12614 LOOP.TF.7.DC:
U 2634, B698.15 :12615 MOV LS[ALTER.ODD] TO WR[0]
U 2635, BF20.15 :12616 MOV WR[0] TO LS[L90] ;PUT DATA IN LS
U 2636, 2F80.15 :12617 CLR WR[0]
U 2637, 3720.15 :12618 MOV LS[L90] TO WR[0]
U 2638, 3698.95 :12619 MOV LS[ALTER.ODD] TO WR[1] ;EXPECTED = SAME
U 2639, 0869.3C :12620 JSR [CHECK.RESULT] ;CHECK MOV.LS.WR
U 263A, 8A63.44 :12621 JMP [LOOP.TF.7.DC] ;LOOP ON ERROR IF ENABLED
U 263B, 8A70.1C :12622 JSR [INC.OTHER]
:12623 LOOP.TF.7.DD:
U 263C, B698.15 :12624 MOV LS[ALTER.ODD] TO WR[0]
U 263D, 3F60.15 :12625 MOV WR[0] TO LS[L80] ;PUT DATA IN LS
U 263E, 2F80.15 :12626 CLR WR[0]
U 263F, B760.15 :12627 MOV LS[L80] TO WR[0]
U 2640, 3698.95 :12628 MOV LS[ALTER.ODD] TO WR[1] ;EXPECTED = SAME
U 2641, 0869.3C :12629 JSR [CHECK.RESULT] ;CHECK MOV.LS.WR
U 2642, 0A63.C4 :12630 JMP [LOOP.TF.7.DD] ;LOOP ON ERROR IF ENABLED
U 2643, 8A70.1C :12631 JSR [INC.OTHER]
:12632 LOOP.TF.7.DE:
U 2644, B698.15 :12633 MOV LS[ALTER.ODD] TO WR[0]
U 2645, 3FA0.15 :12634 MOV WR[0] TO LS[LDO] ;PUT DATA IN LS
U 2646, 2F80.15 :12635 CLR WR[0]
U 2647, B7A0.15 :12636 MOV LS[LDO] TO WR[0]
U 2648, 3698.95 :12637 MOV LS[ALTER.ODD] TO WR[1] ;EXPECTED = SAME
U 2649, 0869.3C :12638 JSR [CHECK.RESULT] ;CHECK MOV.LS.WR
U 264A, 0A64.44 :12639 JMP [LOOP.TF.7.DE] ;LOOP ON ERROR IF ENABLED
U 264B, 8A70.1C :12640 JSR [INC.OTHER]
:12641 LOOP.TF.7.DF:
U 264C, B698.15 :12642 MOV LS[ALTER.ODD] TO WR[0]
U 264D, BFE0.15 :12643 MOV WR[0] TO LS[LFO] ;PUT DATA IN LS
U 264E, 2F80.15 :12644 CLR WR[0]
U 264F, 37E0.15 :12645 MOV LS[LFO] TO WR[0]
U 2650, 3698.95 :12646 MOV LS[ALTER.ODD] TO WR[1] ;EXPECTED = SAME
U 2651, 0869.3C :12647 JSR [CHECK.RESULT] ;CHECK MOV.LS.WR
U 2652, 8A64.C4 :12648 JMP [LOOP.TF.7.DF] ;LOOP ON ERROR IF ENABLED
U 2653, 8A70.1C :12649 JSR [INC.OTHER]
U 2654, 364E.15 :12650 MOV LS[BIT7] TO WR[0]
U 2655, C74C.15 :12651 BIS LS[BIT6] TO WR[0]
U 2656, C74A.15 :12652 BIS LS[BIT5] TO WR[0]
U 2657, 4748.15 :12653 BIS LS[BIT4] TO WR[0]
U 2658, BE88.15 :12654 MOV WR[0] TO LS[ADDRESS.DATA] ;INIT "OTHER" DATA FIELD TO FO
:12655 LOOP.TF.7.FO:
U 2659, 3642.15 :12656 MOV LS[#2] TO WR[0]
U 265A, 3EF8.15 :12657 MOV WR[0] TO LS[OS] ;OS = 2
U 265B, 369F.15 :12658 MOV LS[ONES] TO WR[2] ;WR2 = -1
U 265C, 3C11.15 :12659 ADD OS PLUS WR[2] TO LS[T8]
U 265D, B610.15 :12660 MOV LS[T8] TO WR[0]
U 265E, 3640.95 :12661 MOV LS[#1] TO WR[1] ;EXPECTED = 1
U 265F, 0869.3C :12662 JSR [CHECK.RESULT] ;CHECK WR.PLUS.OS
  
```

```

ZZ-ENKCB-1.0      : ENKCB.MIC
: ENKCB.MCR
: ENKCB.MIC
M 5
TEST F - DEST CTL R 3-MAR-82
MICRO2 1L(03) 3-MAR-82 10:02:46
TEST F - DEST CTL ROM And SRC.ALU CTL PAL Tests (M8390 CPU Module)
Fiche 2 Frame M5
Sequence 270
Page 264
ENKCB Micro-diagnostic for DAP module Rev 01.00

U 2660, 0A65,94 :12663      JMP [LOOP.TF.7.F0]          ;LOOP ON ERROR IF ENABLED
U 2661, 8A70,1C :12664      JSR [INC.OTHER]
:12665      LOOP.TF.7.F1:
U 2662, 3642,15 :12666      MOV LS[2] TO WR[0]
U 2663, 3EF8,15 :12667      MOV WR[0] TO LS[OS]          ;OS = 2
U 2664, 369F,15 :12668      MOV LS[ONES] TO WR[2]        ;WR2 = -1
U 2665, BC61,15 :12669      ADD OS PLUS WR[2] TO LS[L30]
U 2666, 3660,15 :12670      MOV LS[L30] TO WR[0]
U 2667, 3640,95 :12671      MOV LS[1] TO WR[1]          ;EXPECTED = 1
U 2668, 0869,3C :12672      JSR [CHECK.RESULT]          ;CHECK WR.PLUS.OS
U 2669, 8A66,24 :12673      JMP [LOOP.TF.7.F1]          ;LOOP ON ERROR IF ENABLED
U 266A, 8A70,1C :12674      JSR [INC.OTHER]
:12675      LOOP.TF.7.F2:
U 266B, 3642,15 :12676      MOV LS[2] TO WR[0]
U 266C, 3EF8,15 :12677      MOV WR[0] TO LS[OS]          ;OS = 2
U 266D, 369F,15 :12678      MOV LS[ONES] TO WR[2]        ;WR2 = -1
U 266E, BCA7,15 :12679      ADD OS PLUS WR[2] TO LS[L53]
U 266F, 36A6,15 :12680      MOV LS[L53] TO WR[0]
U 2670, 3640,95 :12681      MOV LS[1] TO WR[1]          ;EXPECTED = 1
U 2671, 0869,3C :12682      JSR [CHECK.RESULT]          ;CHECK WR.PLUS.OS
U 2672, 8A66,B4 :12683      JMP [LOOP.TF.7.F2]          ;LOOP ON ERROR IF ENABLED
U 2673, 8A70,1C :12684      JSR [INC.OTHER]
:12685      LOOP.TF.7.F3:
U 2674, 3642,15 :12686      MOV LS[2] TO WR[0]
U 2675, 3EF8,15 :12687      MOV WR[0] TO LS[OS]          ;OS = 2
U 2676, 369F,15 :12688      MOV LS[ONES] TO WR[2]        ;WR2 = -1
U 2677, 3CE7,15 :12689      ADD OS PLUS WR[2] TO LS[L73]
U 2678, B6E6,15 :12690      MOV LS[L73] TO WR[0]
U 2679, 3640,95 :12691      MOV LS[1] TO WR[1]          ;EXPECTED = 1
U 267A, 0869,3C :12692      JSR [CHECK.RESULT]          ;CHECK WR.PLUS.OS
U 267B, 0A67,44 :12693      JMP [LOOP.TF.7.F3]          ;LOOP ON ERROR IF ENABLED
U 267C, 8A70,1C :12694      JSR [INC.OTHER]
U 267D, B62C,15 :12695      MOV LS[FFFFFFF0] TO WR[0]
U 267E, 3E8A,15 :12696      MOV WR[0] TO LS[ERROR.MASK]    ;IGNORE SIGN EXTEND OF OS
:12697      LOOP.TF.7.F4:
U 267F, 3642,15 :12698      MOV LS[2] TO WR[0]
U 2680, 3EF8,15 :12699      MOV WR[0] TO LS[OS]          ;OS = 2
U 2681, 369F,15 :12700      MOV LS[ONES] TO WR[2]        ;WR2 = -1
U 2682, BD21,15 :12701      ADD OS PLUS WR[2] TO LS[L90]
U 2683, 3720,15 :12702      MOV LS[L90] TO WR[0]
U 2684, 3640,95 :12703      MOV LS[1] TO WR[1]          ;EXPECTED = 1
U 2685, 0869,3C :12704      JSR [CHECK.RESULT]          ;CHECK WR.PLUS.OS
U 2686, 8A67,F4 :12705      JMP [LOOP.TF.7.F4]          ;LOOP ON ERROR IF ENABLED
U 2687, 8A70,1C :12706      JSR [INC.OTHER]
:12707      LOOP.TF.7.F5:
U 2688, 3642,15 :12708      MOV LS[2] TO WR[0]
U 2689, 3EF8,15 :12709      MOV WR[0] TO LS[OS]          ;OS = 2
U 268A, 369F,15 :12710      MOV LS[ONES] TO WR[2]        ;WR2 = -1
U 268B, 3D61,15 :12711      ADD OS PLUS WR[2] TO LS[L80]
U 268C, B760,15 :12712      MOV LS[L80] TO WR[0]
U 268D, 3640,95 :12713      MOV LS[1] TO WR[1]          ;EXPECTED = 1
U 268E, 0869,3C :12714      JSR [CHECK.RESULT]          ;CHECK WR.PLUS.OS
U 268F, 0A68,84 :12715      JMP [LOOP.TF.7.F5]          ;LOOP ON ERROR IF ENABLED
U 2690, 8A70,1C :12716      JSR [INC.OTHER]
:12717      LOOP.TF.7.F6:

```



```

U 2691. 3642.15 :12718      MOV LS[#2] TO WR[0]
U 2692. 3EF8.15 :12719      MOV WR[0] TO LS[OS]           ;OS = 2
U 2693. 369F.15 :12720      MOV LS[ONES] TO WR[2]           ;WR2 = -1
U 2694. 3DA1.15 :12721      ADD OS PLUS WR[2] TO LS[LDO]
U 2695. B7A0.15 :12722      MOV LS[LDO] TO WR[0]
U 2696. 3640.95 :12723      MOV LS[#1] TO WR[1]           ;EXPECTED = 1
U 2697. 0869.3C :12724      JSR [CHECK.RESULT]           ;CHECK WR.PLUS.OS
U 2698. 8A69.14 :12725      JMP [LOOP.TF.7.F6]           ;LOOP ON ERROR IF ENABLED
U 2699. 8A70.1C :12726      JSR [INC.OTHER]
:12727      LOOP.TF.7.F7:
U 269A. 3642.15 :12728      MOV LS[#2] TO WR[0]
U 269B. 3EF8.15 :12729      MOV WR[0] TO LS[OS]           ;OS = 2
U 269C. 369F.15 :12730      MOV LS[ONES] TO WR[2]           ;WR2 = -1
U 269D. BDE1.15 :12731      ADD OS PLUS WR[2] TO LS[LFO]
U 269E. 37E0.15 :12732      MOV LS[LFO] TO WR[0]
U 269F. 3640.95 :12733      MOV LS[#1] TO WR[1]           ;EXPECTED = 1
U 26A0. 0869.3C :12734      JSR [CHECK.RESULT]           ;CHECK WR.PLUS.OS
U 26A1. 0A69.A4 :12735      JMP [LOOP.TF.7.F7]           ;LOOP ON ERROR IF ENABLED
U 26A2. 8A70.1C :12736      JSR [INC.OTHER]
U 26A3. E58A.15 :12737      CLR LS[ERROR.MASK]           ;RESTORE MASK
:12738      LOOP.TF.7.F8:
U 26A4. E510.15 :12739      CLR LS[T8]
U 26A5. 3699.15 :12740      MOV LS[ALTER.ODD] TO WR[2]       ;PUT DATA IN WR
U 26A6. BE11.15 :12741      MOV WR[2] TO LS[T8]
U 26A7. B610.15 :12742      MOV LS[T8] TO WR[0]
U 26A8. 3698.95 :12743      MOV LS[ALTER.ODD] TO WR[1]       ;EXPECTED = SAME
U 26A9. 0869.3C :12744      JSR [CHECK.RESULT]           ;CHECK MOV.WR.LS
U 26AA. 8A6A.44 :12745      JMP [LOOP.TF.7.F8]           ;LOOP ON ERROR IF ENABLED
U 26AB. 8A70.1C :12746      JSR [INC.OTHER]
:12747      LOOP.TF.7.F9:
U 26AC. 6560.15 :12748      CLR LS[L30]
U 26AD. 3699.15 :12749      MOV LS[ALTER.ODD] TO WR[2]       ;PUT DATA IN WR
U 26AE. 3E61.15 :12750      MOV WR[2] TO LS[L30]
U 26AF. 3660.15 :12751      MOV LS[L30] TO WR[0]
U 26B0. 3698.95 :12752      MOV LS[ALTER.ODD] TO WR[1]       ;EXPECTED = SAME
U 26B1. 0869.3C :12753      JSR [CHECK.RESULT]           ;CHECK MOV.WR.LS
U 26B2. 0A6A.C4 :12754      JMP [LOOP.TF.7.F9]           ;LOOP ON ERROR IF ENABLED
U 26B3. 8A70.1C :12755      JSR [INC.OTHER]
:12756      LOOP.TF.7.FA:
U 26B4. 65A6.15 :12757      CLR LS[L53]
U 26B5. 3699.15 :12758      MOV LS[ALTER.ODD] TO WR[2]       ;PUT DATA IN WR
U 26B6. 3EA7.15 :12759      MOV WR[2] TO LS[L53]
U 26B7. 36A6.15 :12760      MOV LS[L53] TO WR[0]
U 26B8. 3698.95 :12761      MOV LS[ALTER.ODD] TO WR[1]       ;EXPECTED = SAME
U 26B9. 0869.3C :12762      JSR [CHECK.RESULT]           ;CHECK MOV.WR.LS
U 26BA. 0A6B.44 :12763      JMP [LOOP.TF.7.FA]           ;LOOP ON ERROR IF ENABLED
U 26BB. 8A70.1C :12764      JSR [INC.OTHER]
:12765      LOOP.TF.7.FB:
U 26BC. E5E6.15 :12766      CLR LS[L73]
U 26BD. 3699.15 :12767      MOV LS[ALTER.ODD] TO WR[2]       ;PUT DATA IN WR
U 26BE. BEE7.15 :12768      MOV WR[2] TO LS[L73]
U 26BF. B6E6.15 :12769      MOV LS[L73] TO WR[0]
U 26C0. 3698.95 :12770      MOV LS[ALTER.ODD] TO WR[1]       ;EXPECTED = SAME
U 26C1. 0869.3C :12771      JSR [CHECK.RESULT]           ;CHECK MOV.WR.LS
U 26C2. 8A6B.C4 :12772      JMP [LOOP.TF.7.FB]           ;LOOP ON ERROR IF ENABLED
  
```

```

U 26C3, 8A70,1C :12773 JSR [INC.OTHER]
:12774 LOOP.TF.7.FC:
U 26C4, 2F85,15 :12775 CLR WR[2]
U 26C5, 3F21,15 :12776 MOV WR[2] TO LS[L90]
U 26C6, 3699,15 :12777 MOV LS[ALTER.ODD] TO WR[2] ;PUT DATA IN WR
U 26C7, 3F21,15 :12778 MOV WR[2] TO LS[L90]
U 26C8, 3720,15 :12779 MOV LS[L90] TO WR[0]
U 26C9, 3698,95 :12780 MOV LS[ALTER.ODD] TO WR[1] ;EXPECTED = SAME
U 26CA, 0869,3C :12781 JSR [CHECK.RESULT] ;CHECK MOV.WR.LS
U 26CB, 8A6C,44 :12782 JMP [LOOP.TF.7.FC] ;LOOP ON ERROR IF ENABLED
U 26CC, 8A70,1C :12783 JSR [INC.OTHER]
:12784 LOOP.TF.7.FD:
U 26CD, 2F85,15 :12785 CLR WR[2]
U 26CE, BF61,15 :12786 MOV WR[2] TO LS[L80]
U 26CF, 3699,15 :12787 MOV LS[ALTER.ODD] TO WR[2] ;PUT DATA IN WR
U 26D0, BF61,15 :12788 MOV WR[2] TO LS[L80]
U 26D1, B760,15 :12789 MOV LS[L80] TO WR[0]
U 26D2, 3698,95 :12790 MOV LS[ALTER.ODD] TO WR[1] ;EXPECTED = SAME
U 26D3, 0869,3C :12791 JSR [CHECK.RESULT] ;CHECK MOV.WR.LS
U 26D4, 8A6C,D4 :12792 JMP [LOOP.TF.7.FD] ;LOOP ON ERROR IF ENABLED
U 26D5, 8A70,1C :12793 JSR [INC.OTHER]
:12794 LOOP.TF.7.FE:
U 26D6, 2F85,15 :12795 CLR WR[2]
U 26D7, BFA1,15 :12796 MOV WR[2] TO LS[L00]
U 26D8, 3699,15 :12797 MOV LS[ALTER.ODD] TO WR[2] ;PUT DATA IN WR
U 26D9, BFA1,15 :12798 MOV WR[2] TO LS[L00]
U 26DA, B7A0,15 :12799 MOV LS[L00] TO WR[0]
U 26DB, 3698,95 :12800 MOV LS[ALTER.ODD] TO WR[1] ;EXPECTED = SAME
U 26DC, 0869,3C :12801 JSR [CHECK.RESULT] ;CHECK MOV.WR.LS
U 26DD, 8A6D,64 :12802 JMP [LOOP.TF.7.FE] ;LOOP ON ERROR IF ENABLED
U 26DE, 8A70,1C :12803 JSR [INC.OTHER]
:12804 LOOP.TF.7.FF:
U 26DF, 2F85,15 :12805 CLR WR[2]
U 26E0, 3FE1,15 :12806 MOV WR[2] TO LS[LFO]
U 26E1, 3699,15 :12807 MOV LS[ALTER.ODD] TO WR[2] ;PUT DATA IN WR
U 26E2, 3FE1,15 :12808 MOV WR[2] TO LS[LFO]
U 26E3, 37E0,15 :12809 MOV LS[LFO] TO WR[0]
U 26E4, 3698,95 :12810 MOV LS[ALTER.ODD] TO WR[1] ;EXPECTED = SAME
U 26E5, 0869,3C :12811 JSR [CHECK.RESULT] ;CHECK MOV.WR.LS
U 26E6, 8A6D,F4 :12812 JMP [LOOP.TF.7.FF] ;LOOP ON ERROR IF ENABLED
:12813 TF.RESTORE.LS:
U 26E7, 3612,15 :12814 MOV LS[T9] TO WR[0] ;RESTORE LS LOCATIONS USED IN PARTS 6&7
U 26E8, B614,95 :12815 MOV LS[T10] TO WR[1]
U 26E9, 3617,15 :12816 MOV LS[T11] TO WR[2]
U 26EA, BE60,15 :12817 MOV WR[0] TO LS[L30]
U 26EB, 3EA6,95 :12818 MOV WR[1] TO LS[L53]
U 26EC, BEE7,15 :12819 MOV WR[2] TO LS[L73]
U 26ED, 3618,15 :12820 MOV LS[T12] TO WR[0]
U 26EE, 361A,95 :12821 MOV LS[T13] TO WR[1]
U 26EF, 361D,15 :12822 MOV LS[T14] TO WR[2]
U 26F0, 361F,95 :12823 MOV LS[T15] TO WR[3]
U 26F1, BF20,15 :12824 MOV WR[0] TO LS[L90]
U 26F2, BF60,95 :12825 MOV WR[1] TO LS[L80]
U 26F3, BFA1,15 :12826 MOV WR[2] TO LS[L00]
U 26F4, BFE1,95 :12827 MOV WR[3] TO LS[LFO]

```

```

U 26F5, 8A70,54 :12828      JMP [END.TF]
                  :12829
U 26F6, A006,15 :12830      CHECK.WR3:
U 26F7, B699,95 :12831      MOV WR[3] TO WR[0]          ;RECEIVED DATA
U 26F8, 2006,95 :12832      MOV LS[ALTER.ODD] TO WR[3]      ;RESTORE WR3 IN CASE IT WAS WRITTEN
U 26F9, 0869,3C :12833      MOV WR[3] TO WR[1]          ;EXPECTED DATA
U 26FA, 5800,14 :12834      JSR [CHECK.RESULT]            ;CHECK
U 26FB, DB00,16 :12835      RETURN                          ;LOOP ON ERROR IF ENABLED
                  :12836      RETURN+1                      ;"NORMAL" RETURN
U 26FC, B69A,95 :12837      CHECK.WR0:
U 26FD, 0869,3C :12838      MOV LS[ALTER.EVEN] TO WR[1]      ;EXPECTED DATA
U 26FE, 5800,14 :12839      JSR [CHECK.RESULT]            ;CHECK
U 26FF, 369A,15 :12840      RETURN                          ;LOOP ON ERROR IF ENABLED
U 2700, DB00,16 :12841      MOV LS[ALTER.EVEN] TO WR[0]      ;RESTORE WR0 IN CASE IT WAS WRITTEN
                  :12842      RETURN+1                      ;"NORMAL" RETURN
U 2701, B688,95 :12843      INC.OTHER:
U 2702, 2042,95 :12844      MOV LS[ADDRESS.DATA] TO WR[1]
U 2703, 3E88,95 :12845      INC WR[1]
U 2704, 5800,14 :12846      MOV WR[1] TO LS[ADDRESS.DATA]    ;INCREMENT "OTHER" DATA
                  :12847      RETURN
U 2705, 365C,15 :12848      END.TF:
U 2706, 3E80,15 :12849      END.SECTION:
                  :12850      MOV LS[BIT14] TO WR[0]          ;SET UP WR 0 FOR END OF SECTION
                  :12851      MOV WR[0] TO LS[CONTROL.STATUS] ;SET UP CONTROL AND STATUS WORD TO
                  :12852                                  ;REPORT END OF SECTION
U 2707, 10E0,15 :12853      MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
U 2708, 0A70,84 :12854      WAIT.EOS:
                  :12855      JMP [WAIT.EOS]
                  :12856                                  ;LOOP HERE WAITING FOR CONSOLE TO
                  :                                  ;RESPOND
  
```

ZZ-ENKCB-1.0 ;
: ENKCB.MCR
:

Cross Reference Lis 3-MAR-82 D 6
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Sequence 274
Cross Reference Listing - Field Names and Defined Values Rev 01.00 Page 268

ZZ-ENKCB-1.0
; ENKCB.MCR
;

MICRO2 1L(03) Cross Reference Lis 3-MAR-82 F 6
Cross Reference Listing - Macro Names ENKCB Micro-diagnostic for DAP module Sequence 275
Rev 01.00 Page 269

ZZ-ENKCB-1.0 ;
: ENKCB.MCR
:

Cross Reference Lis 3-MAR-82 F 6
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Sequence 276
Cross Reference Listing - Expression Names Rev 01.00 Page 270

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
C 0000	1772:	1773:	1774:	1775:	1776:	1777:	1778:	1779:
C 0008	1780:	1781:	1782:	1783:	1784:	1785:	1786:	1787:
C 0010	1791:	1792:	1793:	1794:	1795:	1796:	1797:	1798:
C 0018	1799:	1800:	1801:	1802:	1803:	1804:	1805:	1806:
C 0020	1815:	1816:	1817:	1818:	1819:	1820:	1821:	1822:
C 0028	1823:	1824:	1825:	1826:	1827:	1828:	1829:	1830:
C 0030	1831:	1832:	1833:	1834:	1835:	1836:	1837:	1838:
C 0038	1839:	1840:	1841:	1842:	1843:	1844:	1845:	1846:
C 0040	1855:	1856:	1857:	1858:	1859:	1860:	1861:	1862:
C 0048	1863:	1864:	1865:	1866:	1867:	1868:	1869:	1870:
C 0050	1871:	1872:	1873:	1874:	1875:	1876:	1877:	1878:
C 0058	1879:	1880:	1881:	1882:	1883:	1884:	1885:	1886:
C 0060	1894:	1895:	1896:	1897:	1898:	1899:	1900:	1901:
C 0068	1902:	1903:	1904:	1905:	1906:	1907:	1908:	1909:
C 0070	1910:	1911:	1912:	1913:	1914:	1915:	1916:	1917:
C 0078	1918:	1919:	1920:	1921:	1922:	1923:	1924:	1925:
C 0080	1932:	1934:	1936:	1938:	1941:	1943:	1945:	1947:
C 0088	1950:	1952:	1954:	1956:	1959:	1961:	1963:	1965:
C 0090	1968:	1970:	1972:	1974:	1977:	1979:	1981:	1983:
C 0098	1986:	1988:	1990:	1992:	1995:	1997:	1999:	2001:
C 00A0	2004:	2006:	2008:	2010:	2013:	2015:	2017:	2019:
C 00A8	2022:	2024:	2026:	2028:	2031:	2033:	2035:	2037:
C 00B0	2040:	2042:	2044:	2046:	2049:	2051:	2053:	2055:
C 00B8	2058:	2060:	2062:	2064:	2067:	2069:	2071:	2073:
C 00C0	2081:	2082:	2083:	2084:	2085:	2086:	2087:	2088:
C 00C8	2091:	2092:	2093:	2094:	2095:	2096:	2097:	2098:
C 00D0	2101:	2102:	2103:	2104:	2105:	2106:	2107:	2108:
C 00D8	2111:	2112:	2113:	2114:	2115:	2116:	2117:	2118:
C 00E0	2122:	2123:	2124:	2125:	2126:	2127:	2128:	2129:
C 00E8	2131:	2132:	2133:	2134:	2135:	2136:	2137:	2138:
C 00F0	2142:	2143:	2144:	2145:	2146:	2147:	2148:	2149:
C 00F8	2152:	2153:	2154:	2155:	2156:	2157:	2158:	2159:
C 0100	2167:	2168:	2169:	2170:	2172:	2173:	2174:	2175:
C 0108	2177:	2178:	2179:	2180:	2182:	2183:	2184:	2185:
C 0110	2188:	2189:	2190:	2191:	2193:	2194:	2195:	2196:
C 0118	2198:	2199:	2200:	2201:	2203:	2204:	2205:	2206:
C 0120	2209:	2210:	2211:	2212:	2214:	2215:	2216:	2217:
C 0128	2219:	2220:	2221:	2222:	2224:	2225:	2226:	2227:
C 0130	2230:	2231:	2232:	2233:	2235:	2236:	2237:	2238:
C 0138	2240:	2241:	2242:	2243:	2245:	2246:	2247:	2248:
C 0140	2251:	2252:	2253:	2254:	2256:	2257:	2258:	2259:
C 0148	2261:	2262:	2263:	2264:	2266:	2267:	2268:	2269:
C 0150	2272:	2273:	2274:	2275:	2277:	2278:	2279:	2280:
C 0158	2282:	2283:	2284:	2285:	2287:	2288:	2289:	2290:
C 0160	2293:	2294:	2295:	2296:	2298:	2299:	2300:	2301:
C 0168	2303:	2304:	2305:	2306:	2308:	2309:	2310:	2311:
C 0170	2314:	2315:	2316:	2317:	2319:	2320:	2321:	2322:
C 0178	2324:	2325:	2326:	2327:	2329:	2330:	2331:	2332:
C 0180	2342:	2343:	2344:	2345:	2347:	2348:	2349:	2350:
C 0188	2352:	2353:	2354:	2355:	2357:	2358:	2359:	2360:
C 0190	2363:	2364:	2365:	2366:	2368:	2369:	2370:	2371:
C 0198	2373:	2374:	2375:	2376:	2378:	2379:	2380:	2381:
C 01A0	2384:	2385:	2386:	2387:	2389:	2390:	2391:	2392:

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
C 01A8	2394:	2395:	2396:	2397:	2399:	2400:	2401:	2402:
C 01B0	2405:	2406:	2407:	2408:	2410:	2411:	2412:	2413:
C 01B8	2415:	2416:	2417:	2418:	2420:	2421:	2422:	2423:
C 01C0	2426:	2427:	2428:	2429:	2431:	2432:	2433:	2434:
C 01C8	2436:	2437:	2438:	2439:	2441:	2442:	2443:	2444:
C 01D0	2447:	2448:	2449:	2450:	2452:	2453:	2454:	2455:
C 01D8	2457:	2458:	2459:	2460:	2462:	2463:	2464:	2465:
C 01E0	2468:	2469:	2470:	2471:	2473:	2474:	2475:	2476:
C 01E8	2478:	2479:	2480:	2481:	2483:	2484:	2485:	2486:
C 01F0	2489:	2490:	2491:	2492:	2494:	2495:	2496:	2497:
C 01F8	2499:	2500:	2501:	2502:	2504:	2505:	2506:	2507:

ZZ-ENKCB-1.0
: ENKCB.MCR
:

Location / Line Num 3-MAR-82
MICRO2 1L(03) 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module
Location / Line Number Index

Fiche 2 Frame 16

Sequence 279

Rev 01.00 Page 273

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 0000	4038:	3874:	3875:	3876:	3877:	3878:	3879:	3880:
U 0008	3881:	3882:	3883:	3884:	3885:	3886:	3887:	3888:
U 0010	3889:	3891:	3893:	3895:	3897:	3899:	3901:	3903:
U 0018	3905:	3907:	3909:	3911:	3913:	3915:	3917:	3919:
U 0020	3921:	3923:	3925:	3927:	3929:	3931:	3933:	3935:
U 0028	3937:	3939:	3941:	3943:	3945:	3947:	3949:	3951:
U 0030	3953:	3955:	3957:	3959:	3961:	3963:	3965:	3967:
U 0038	3969:	3971:	3973:	3975:	3977:	3979:	3981:	3983:
U 0040	3985:	3987:	3989:	3991:	3993:	3995:	3997:	3999:
U 0048	4001:	4003:	4005:	4007:	4009:	4011:	4013:	4015:
U 0050	4047:	4049:	4051:	4054:	4057:	4061:	4063:	4065:
U 0058	4067:	4069:	4071:	4073:	4075:	4077:	4079:	4081:
U 0060	4083:	4085:	4087:	4089:	4091:	4093:	4094:	4095:
U 0068	4096:	4097:	4098:	4099:	4100:	4101:	4102:	4103:
U 0070	4133:	4135:	4137:	4138:	4140:	4141:	4143:	4144:
U 0078	4146:	4147:	4149:	4150:	4152:	4153:	4155:	4156:
U 0080	4158:	4159:	4161:	4162:	4164:	4165:	4167:	4168:
U 0088	4170:	4171:	4173:	4174:	4176:	4177:	4179:	4180:
U 0090	4182:	4183:	4185:	4186:	4188:	4189:	4191:	4192:
U 0098	4194:	4195:	4197:	4198:	4200:	4201:	4203:	4204:
U 00A0	4206:	4207:	4209:	4210:	4212:	4213:	4215:	4216:
U 00A8	4218:	4219:	4221:	4222:	4224:	4225:	4227:	4228:
U 00B0	4230:	4231:	4233:	4234:	4236:	4237:	4239:	4240:
U 00B8	4242:	4243:	4245:	4246:	4248:	4249:	4251:	4252:
U 00C0	4254:	4255:	4257:	4258:	4260:	4261:	4263:	4264:
U 00C8	4266:	4267:	4269:	4270:	4272:	4273:	4275:	4276:
U 00D0	4278:	4279:	4281:	4282:	4284:	4285:	4287:	4288:
U 00D8	4290:	4291:	4293:	4294:	4296:	4297:	4299:	4300:
U 00E0	4302:	4303:	4305:	4306:	4308:	4309:	4311:	4312:
U 00E8	4314:	4315:	4317:	4318:	4320:	4321:	4323:	4324:
U 00F0	4326:	4327:	4329:	4330:	4332:	4333:	4335:	4336:
U 00F8	4338:	4339:	4341:	4342:	4344:	4345:	4347:	4348:
U 0100	4350:	4351:	4353:	4354:	4356:	4357:	4359:	4360:
U 0108	4362:	4363:	4365:	4366:	4368:	4369:	4371:	4372:
U 0110	4374:	4375:	4377:	4378:	4380:	4381:	4383:	4384:
U 0118	4386:	4387:	4389:	4390:	4392:	4393:	4395:	4396:
U 0120	4398:	4399:	4401:	4402:	4404:	4405:	4407:	4408:
U 0128	4410:	4411:	4413:	4414:	4416:	4417:	4419:	4420:
U 0130	4422:	4423:	4425:	4426:	4428:	4429:	4431:	4432:
U 0138	4434:	4435:	4437:	4438:	4440:	4441:	4443:	4444:
U 0140	4446:	4447:	4449:	4450:	4452:	4453:	4455:	4456:
U 0148	4458:	4459:	4461:	4462:	4464:	4465:	4467:	4468:
U 0150	4470:	4471:	4473:	4474:	4476:	4477:	4479:	4480:
U 0158	4482:	4483:	4485:	4486:	4488:	4489:	4491:	4492:
U 0160	4494:	4495:	4504:	4506:	4508:	4509:	4511:	4512:
U 0168	4514:	4515:	4517:	4518:	4520:	4521:	4523:	4524:
U 0170	4526:	4527:	4529:	4530:	4532:	4533:	4535:	4536:
U 0178	4538:	4539:	4541:	4542:	4544:	4545:	4547:	4548:
U 0180	4550:	4551:	4553:	4554:	4556:	4557:	4559:	4560:
U 0188	4562:	4563:	4565:	4566:	4568:	4569:	4571:	4572:
U 0190	4574:	4575:	4577:	4578:	4580:	4581:	4583:	4584:
U 0198	4586:	4587:	4589:	4590:	4592:	4593:	4595:	4596:
U 01A0	4598:	4599:	4601:	4602:	4604:	4605:	4607:	4608:

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 01A8	4610:	4611:	4613:	4614:	4616:	4617:	4619:	4620:
U 01B0	4622:	4623:	4625:	4626:	4628:	4629:	4631:	4632:
U 01B8	4634:	4635:	4637:	4638:	4640:	4641:	4643:	4644:
U 01C0	4646:	4647:	4649:	4650:	4652:	4653:	4655:	4656:
U 01C8	4658:	4659:	4661:	4662:	4664:	4665:	4667:	4668:
U 01D0	4670:	4671:	4673:	4674:	4676:	4677:	4679:	4680:
U 01D8	4682:	4683:	4685:	4686:	4688:	4689:	4691:	4692:
U 01E0	4694:	4695:	4697:	4698:	4700:	4701:	4703:	4704:
U 01E8	4706:	4707:	4709:	4710:	4712:	4713:	4715:	4716:
U 01F0	4718:	4719:	4721:	4722:	4724:	4725:	4727:	4728:
U 01F8	4730:	4731:	4733:	4734:	4736:	4737:	4739:	4740:
U 0200	4742:	4743:	4745:	4746:	4748:	4749:	4751:	4752:
U 0208	4754:	4755:	4757:	4758:	4760:	4761:	4763:	4764:
U 0210	4766:	4767:	4769:	4770:	4772:	4773:	4775:	4776:
U 0218	4778:	4779:	4781:	4782:	4784:	4785:	4787:	4788:
U 0220	4790:	4791:	4793:	4794:	4796:	4797:	4799:	4800:
U 0228	4802:	4803:	4805:	4806:	4808:	4809:	4811:	4812:
U 0230	4814:	4815:	4817:	4818:	4820:	4821:	4823:	4824:
U 0238	4826:	4827:	4829:	4830:	4832:	4833:	4835:	4836:
U 0240	4838:	4839:	4841:	4842:	4844:	4845:	4847:	4848:
U 0248	4850:	4851:	4853:	4854:	4856:	4857:	4859:	4860:
U 0250	4862:	4863:	4865:	4866:				
U 0258 - 03F7 Unused								
U 03F8								4874:
U 0400 - 05FF Unused								
U 0600	4896:	4897:	4899:	4902:	4903:	4904:	4918:	4919:
U 0608	4979:	4981:	4983:	4985:	4986:	5041:	5042:	5044:
U 0610	5046:	5048:	5051:	5053:	5054:	5055:	5059:	5060:
U 0618	5061:	5065:	5066:	5067:	5069:	5072:	5073:	5128:
U 0620	5130:	5131:	5134:	5136:	5140:	5142:	5146:	5152:
U 0628	5154:	5155:	5161:	5163:	5165:	5166:	5167:	5173:
U 0630	5174:	5175:	5176:	5177:	5178:	5179:	5182:	5184:
U 0638	5185:	5189:	5239:	5241:	5243:	5245:	5248:	5249:
U 0640	5251:	5252:	5307:	5309:	5310:	5313:	5315:	5319:
U 0648	5321:	5325:	5331:	5333:	5334:	5340:	5341:	5342:
U 0650	5343:	5344:	5345:	5348:	5398:	5400:	5403:	5405:
U 0658	5408:	5409:	5411:	5412:	5460:	5465:	5467:	5469:
U 0660	5470:	5471:	5521:	5523:	5526:	5530:	5532:	5533:
U 0668	5572:	5573:	5574:	5577:	5578:	5579:	5582:	5583:
U 0670	5584:	5587:	5588:	5589:	5592:	5593:	5597:	5598:
U 0678	5599:	5638:	5639:	5640:	5644:	5645:	5646:	5650:
U 0680	5651:	5652:	5656:	5657:	5661:	5662:	5666:	5669:
U 0688	5671:	5718:	5719:	5720:	5721:	5722:	5769:	5770:
U 0690	5771:	5772:	5773:	5841:	5843:	5846:	5847:	5849:
U 0698	5851:	5852:	5853:	5855:	5859:	5861:	5863:	5867:
U 06A0	5868:	5871:	5872:	5874:	5876:	5877:	5879:	5881:
U 06A8	5886:	5887:	5889:	5891:	5892:	5898:	5899:	5902:
U 06B0	5903:	5904:	5909:	5910:	5912:	5913:	5915:	5917:
U 06B8	5918:	5920:	5922:	5924:	5926:	5928:	5935:	5936:
U 06C0	5938:	5941:	5954:	5955:	5957:	5958:	5959:	5960:
U 06C8	5961:	5962:	5963:	5964:	5965:	5969:	5972:	5973:
U 06D0	5974:	5975:	5976:	5977:	5978:	5979:	5980:	5981:
U 06D8	5982:	5983:	5984:	5985:	5986:	5987:	5990:	5993:

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 06E0	5996:	5998:	5999:	6000:	6001:	6004:	6007:	6008:
U 06E8	6011:	6012:	6013:	6014:	6016:	6017:	6018:	6019:
U 06F0	6021:	6023:	6025:	6026:				
U 06F8 - 06FF	Unused							
U 0700	6036:	6037:	6038:	6039:	6043:	6044:	6045:	6046:
U 0708	6050:	6051:	6052:	6056:	6057:	6058:	6059:	6063:
U 0710	6064:	6065:	6070:	6071:	6072:	6073:		
U 0718	6078:	6189:	6190:	6192:	6194:	6195:	6196:	6197:
U 0720	6198:	6199:	6200:	6201:	6202:	6203:	6204:	6206:
U 0728	6207:	6210:	6211:	6212:				
U 0730 - 0817	Unused							
U 0818	6318:	6321:						
U 0820 - 1017	Unused							
U 1018	6324:							
U 1020 - 1417	Unused							
U 1418	6327:							
U 1420 - 14FF	Unused							
U 1500	6289:	6290:	6291:	6292:	6293:	6296:	6297:	6298:
U 1508	6300:	6301:	6303:	6305:	6306:	6307:	6308:	6309:
U 1510	6310:	6312:	6314:					
U 1518 - 151F	Unused							
U 1520	6418:	6419:	6421:	6423:	6424:	6425:	6426:	6427:
U 1528	6428:	6429:	6430:	6431:	6432:	6434:	6435:	6436:
U 1530	6437:	6438:	6439:	6440:	6442:	6443:	6444:	6445:
U 1538	6446:	6447:	6448:	6450:	6451:	6452:	6453:	6455:
U 1540	6456:	6458:	6459:	6460:	6462:	6463:	6464:	6466:
U 1548	6467:	6468:	6470:	6471:	6472:	6474:	6475:	6476:
U 1550	6478:	6479:	6480:	6482:	6483:	6484:	6486:	6487:
U 1558	6488:	6490:	6491:	6492:	6494:	6495:	6496:	6498:
U 1560	6499:	6500:	6502:	6503:	6504:	6506:	6507:	6508:
U 1568	6510:	6511:	6512:					6667:
U 1570	6668:	6670:	6672:	6673:	6674:	6675:	6676:	6677:
U 1578	6678:	6679:	6680:	6681:	6683:	6684:	6685:	6686:
U 1580	6687:	6688:	6689:	6690:	6691:	6692:	6694:	6695:
U 1588	6697:	6698:	6699:	6700:	6701:	6702:	6703:	6705:
U 1590	6706:	6707:	6709:	6710:	6711:	6712:	6713:	6714:
U 1598	6715:	6716:	6717:	6718:	6720:	6721:	6722:	6723:
U 15A0	6725:	6726:	6794:	6795:	6797:	6799:	6800:	6801:
U 15A8	6802:	6803:	6804:	6805:	6806:	6807:	6808:	6810:
U 15B0	6811:	6812:	6813:	6814:	6815:	6816:	6818:	6819:
U 15B8	6820:	6821:	6822:	6823:	6825:	6826:	6827:	6828:
U 15C0	6829:	6830:	6831:	6832:	6834:	6835:	6836:	6837:
U 15C8	6838:	6839:						
U 15D0 - 15DF	Unused							
U 15E0	6842:	6843:	6844:	6845:	6846:	6847:	6848:	6849:
U 15E8	6850:	6851:	6852:	6853:	6854:	6855:	6856:	6857:
U 15F0	6858:	6859:	6860:	6861:	6862:	6863:	6864:	6865:
U 15F8	6866:	6867:	6868:	6869:	6870:	6871:	6872:	6874:
U 1600	6875:	6974:	6975:	6977:	6979:	6980:	6981:	6982:
U 1608	6983:	6984:	6986:	6987:	6988:	6989:	6990:	6991:
U 1610	6992:	6994:	6995:	6996:	6997:	6998:	6999:	7000:
U 1618	7001:	7002:	7003:	7004:	7005:	7007:	7008:	7009:
U 1620	7010:	7011:	7012:	7014:	7015:	7016:	7017:	7018:

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 1628	7019:	7021:	7022:	7023:	7024:	7025:	7026:	7027:
U 1630	7028:	7029:	7030:	7031:	7032:	7034:	7035:	7036:
U 1638	7037:	7038:	7039:	7040:	7042:	7043:	7044:	7045:
U 1640	7046:	7047:	7049:	7050:	7051:	7052:	7054:	7055:
U 1648	7057:	7058:	7060:	7061:	7062:	7063:	7065:	7066:
U 1650	7067:	7069:	7098:	7099:	7301:	7303:	7304:	7305:
U 1658	7306:	7307:	7308:	7309:	7310:	7311:	7312:	7313:
U 1660	7315:	7316:	7317:	7318:	7320:	7321:	7322:	7323:
U 1668	7324:	7325:	7327:	7328:	7329:	7330:	7332:	7333:
U 1670	7334:	7335:	7337:	7338:	7339:	7340:	7341:	7342:
U 1678	7343:	7345:	7346:	7347:	7348:	7350:	7351:	7352:
U 1680	7353:	7355:	7356:	7357:	7358:	7359:	7360:	7361:
U 1688	7362:	7363:	7365:	7366:	7367:	7368:	7369:	7371:
U 1690	7372:	7373:	7374:	7376:	7377:	7378:	7379:	7380:
U 1698	7381:	7382:	7383:	7384:	7385:	7387:	7388:	7389:
U 16A0	7390:	7391:	7393:	7394:	7395:	7396:	7397:	7398:
U 16A8	7399:	7400:	7401:	7402:	7404:	7405:	7406:	7407:
U 16B0	7408:	7409:	7411:	7412:	7413:	7414:	7415:	7416:
U 16B8	7417:	7418:	7420:	7421:	7422:	7423:	7425:	7426:
U 16C0	7427:	7428:	7429:	7430:	7431:	7432:	7433:	7434:
U 16C8	7436:	7437:	7438:	7439:	7440:	7441:	7444:	7445:
U 16D0	7446:	7447:	7449:	7450:	7451:	7452:	7453:	7455:
U 16D8	7456:	7457:	7458:	7460:	7461:	7463:	7464:	7465:
U 16E0	7466:	7467:	7468:	7469:	7470:	7472:	7473:	7474:
U 16E8	7475:	7476:	7477:	7478:	7480:	7481:	7482:	7483:
U 16F0	7484:	7485:	7486:	7487:	7489:	7493:	7494:	7495:
U 16F8	7496:	7497:	7499:	7500:	7501:	7502:	7503:	7504:
U 1700	7505:	7507:	7508:	7509:	7510:	7512:	7513:	7515:
U 1708	7516:	7517:	7518:	7519:	7520:	7521:	7522:	7523:
U 1710	7524:	7525:	7527:	7529:	7530:	7531:	7532:	7533:
U 1718	7535:	7536:	7537:	7538:	7539:	7540:	7541:	7542:
U 1720	7544:	7618:	7619:	7621:	7623:	7624:	7625:	7626:
U 1728	7627:	7628:	7629:	7630:	7631:	7632:	7633:	7634:
U 1730	7635:	7636:	7637:	7638:	7639:	7641:	7642:	7643:
U 1738	7645:	7646:	7647:	7648:	7649:	7650:	7651:	7652:
U 1740	7653:	7654:	7655:	7656:	7657:	7658:	7659:	7661:
U 1748	7663:	7664:	7665:	7666:	7667:	7668:	7669:	7670:
U 1750	7671:	7672:	7673:	7674:	7675:	7676:	7677:	7679:
U 1758	7681:	7682:	7683:	7684:	7685:	7686:	7687:	7688:
U 1760	7689:	7690:	7691:	7693:	7694:	7695:	7696:	7697:
U 1768	7698:	7699:	7700:	7701:	7702:	7703:	7704:	7706:
U 1770	7707:	7708:	7709:	7710:	7711:	7712:	7713:	7714:
U 1778	7715:	7717:	7718:	7719:	7720:	7721:	7722:	7723:
U 1780	7724:	7725:	7726:	7727:	7728:	7729:	7731:	7890:
U 1788	7891:	7893:	7895:	7896:	7897:	7898:	7900:	7901:
U 1790	7902:	7903:	7904:	7905:	7906:	7907:	7908:	7909:
U 1798	7910:	7912:	7913:	7914:	7915:	7916:	7917:	7918:
U 17A0	7919:	7920:	7921:	7923:	7924:	7925:	7926:	7927:
U 17A8	7928:	7930:	7932:	7933:	7934:	7935:	7936:	7937:
U 17B0	7938:	7939:	7940:	7942:	7944:	7945:	7946:	7947:
U 17B8	7948:	7949:	7950:	7951:	7952:	7954:	7956:	7957:
U 17C0	7958:	7959:	7960:	7961:	7962:	7963:	7964:	7966:
U 17C8	7968:	7969:	7970:	7971:	7972:	7973:	7974:	7975:

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 17D0	7976:	7977:	7978:	7979:	7980:	7982:	7983:	7985:
U 17D8	7986:	7987:	7988:	7989:	7990:	7991:	7992:	7993:
U 17E0	7994:	7995:	7996:	7997:	7999:	8000:	8001:	8002:
U 17E8	8003:	8004:	8005:	8006:	8007:	8008:	8009:	8010:
U 17F0	8011:	8013:	8014:	8015:	8016:	8018:	8019:	8020:
U 17F8	8021:	8022:	8023:	8024:	8025:	8027:	8028:	8029:
U 1800	8030:	8032:	8034:	8035:	8036:	8037:	8038:	8039:
U 1808	8040:	8041:	8042:	8043:	8045:	8046:	8047:	8049:
U 1810	8050:	8051:	8053:	8055:	8056:	8057:	8058:	8061:
U 1818	8062:	8063:	8064:	8065:	8066:	8067:	8068:	8071:
U 1820	8072:	8073:	8074:	8075:	8076:	8077:	8078:	8081:
U 1828	8082:	8083:	8084:	8085:	8086:	8087:	8088:	8092:
U 1830	8093:	8094:	8095:	8096:	8098:	8161:	8162:	8164:
U 1838	8166:	8167:	8168:	8169:	8170:	8171:	8172:	8174:
U 1840	8175:	8176:	8177:	8178:	8179:	8180:	8182:	8183:
U 1848	8184:	8185:	8186:	8187:	8188:	8190:	8191:	8192:
U 1850	8193:	8194:	8318:	8319:	8321:	8323:	8324:	8325:
U 1858	8326:	8327:	8329:	8330:	8331:	8332:	8333:	8334:
U 1860	8335:	8336:	8337:	8338:	8339:	8340:	8342:	8343:
U 1868	8344:	8345:	8347:	8348:	8349:	8350:	8351:	8352:
U 1870	8353:	8354:	8355:	8356:	8358:	8359:	8360:	8361:
U 1878	8363:	8364:	8365:	8366:	8367:	8368:	8369:	8370:
U 1880	8371:	8372:	8373:	8374:	8376:	8377:	8378:	8379:
U 1888	8381:	8382:	8383:	8384:	8385:	8386:	8387:	8388:
U 1890	8389:	8390:	8391:	8393:	8394:	8395:	8396:	8398:
U 1898	8399:	8400:	8401:	8402:	8403:	8404:	8405:	8406:
U 18A0	8407:	8408:	8410:	8411:	8412:	8413:	8471:	8472:
U 18A8	8474:	8476:	8477:	8478:	8479:	8480:	8482:	8483:
U 18B0	8484:	8485:	8486:	8487:	8488:	8489:	8490:	8491:
U 18B8	8492:	8494:	8495:	8496:	8497:	8499:	8500:	8501:
U 18C0	8502:	8503:	8504:	8505:	8506:	8507:	8508:	8510:
U 18C8	8511:	8512:	8513:	8690:	8691:	8693:	8695:	8696:
U 18D0	8697:	8698:	8699:	8701:	8702:	8703:	8704:	8705:
U 18D8	8706:	8707:	8708:	8709:	8710:	8711:	8712:	8713:
U 18E0	8715:	8716:	8717:	8718:	8720:	8721:	8722:	8723:
U 18E8	8724:	8725:	8726:	8727:	8728:	8729:	8730:	8731:
U 18F0	8733:	8734:	8735:	8736:	8738:	8739:	8740:	8741:
U 18F8	8742:	8743:	8744:	8745:	8746:	8747:	8748:	8749:
U 1900	8750:	8751:	8752:	8753:	8755:	8756:	8757:	8758:
U 1908	8760:	8761:	8762:	8763:	8764:	8765:	8766:	8767:
U 1910	8768:	8769:	8770:	8771:	8772:	8774:	8775:	8776:
U 1918	8777:	8779:	8780:	8781:	8782:	8783:	8784:	8785:
U 1920	8786:	8787:	8788:	8789:	8790:	8791:	8793:	8794:
U 1928	8795:	8796:	8798:	8799:	8800:	8801:	8802:	8803:
U 1930	8804:	8805:	8806:	8807:	8808:	8809:	8810:	8812:
U 1938	8813:	8814:	8815:	8899:	8900:	8902:	8904:	8905:
U 1940	8906:	8907:	8908:	8910:	8911:	8912:	8913:	8914:
U 1948	8915:	8916:	8917:	8918:	8919:	8920:	8921:	8922:
U 1950	8924:	8925:	8926:	8927:	8929:	8930:	8931:	8932:
U 1958	8933:	8934:	8935:	8936:	8937:	8938:	8939:	8940:
U 1960	8941:	8942:	8943:	8945:	8946:	8947:	8948:	9005:
U 1968	9006:	9008:	9010:	9011:	9012:	9013:	9014:	9015:
U 1970	9016:	9018:	9019:	9020:	9021:	9022:	9023:	9025:

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 1978	9026:	9027:	9028:	9029:	9030:	9031:	9079:	9080:
U 1980	9082:	9084:	9085:	9086:	9087:	9088:	9089:	9090:
U 1988	9091:	9096:	9097:	9098:	9100:	9101:	9102:	9103:
U 1990	9105:	9106:	9107:	9108:	9110:	9111:	9112:	9113:
U 1998	9115:	9116:	9117:	9118:	9120:	9121:	9122:	9123:
U 19A0	9125:	9126:	9127:	9128:	9130:	9131:	9132:	9133:
U 19A8	9135:	9136:	9137:	9138:	9140:	9141:	9142:	9143:
U 19B0	9145:	9146:	9147:	9148:	9150:	9151:	9152:	9153:
U 19B8	9155:	9156:	9157:	9158:	9160:	9161:	9162:	9163:
U 19C0	9165:	9166:	9167:	9168:	9170:	9171:	9172:	9173:
U 19C8	9175:	9176:	9177:	9179:	9180:	9181:	9182:	9183:
U 19D0	9185:	9186:	9187:	9188:	9190:	9191:	9192:	9193:
U 19D8	9195:	9196:	9197:	9198:	9200:	9201:	9202:	9203:
U 19E0	9205:	9206:	9207:	9208:	9210:	9211:	9212:	9213:
U 19E8	9215:	9216:	9217:	9218:	9220:	9221:	9222:	9223:
U 19F0	9225:	9226:	9227:	9228:	9230:	9231:	9232:	9233:
U 19F8	9235:	9236:	9237:	9238:	9240:	9241:	9242:	9243:
U 1A00	9245:	9246:	9247:	9248:	9250:	9251:	9252:	9253:
U 1A08	9255:	9256:	9257:	9258:	9260:	9261:	9262:	9263:
U 1A10 - 1A17 Unused								
U 1A18	6330:		9270:	9271:	9272:	9273:	9274:	9276:
U 1A20	9277:	9278:	9279:	9281:	9282:	9283:	9284:	9286:
U 1A28	9287:	9288:	9289:	9291:	9292:	9293:	9296:	9297:
U 1A30	9298:	9299:	9300:	9301:	9303:	9304:	9305:	9306:
U 1A38	9308:	9309:	9310:	9311:	9313:	9314:	9315:	9316:
U 1A40	9318:	9319:	9320:	9321:	9323:	9324:	9325:	9326:
U 1A48	9328:	9329:	9330:	9331:	9333:	9334:	9335:	9336:
U 1A50	9338:	9339:	9340:	9341:	9343:	9344:	9345:	9346:
U 1A58	9348:	9349:	9350:	9351:	9353:	9354:	9355:	9356:
U 1A60	9358:	9359:	9360:	9361:	9363:	9364:	9365:	9366:
U 1A68	9368:	9369:	9370:	9371:	9373:	9374:	9375:	9376:
U 1A70	9378:	9379:	9380:	9385:	9386:	9387:	9389:	9390:
U 1A78	9391:	9392:	9393:	9394:	9395:	9397:	9398:	9399:
U 1A80	9400:	9401:	9402:	9404:	9405:	9406:	9407:	9408:
U 1A88	9409:	9410:	9411:	9413:	9414:	9415:	9416:	9417:
U 1A90	9418:	9419:	9420:	9421:	9422:	9424:	9425:	9426:
U 1A98	9427:	9428:	9429:	9430:	9431:	9432:	9433:	9435:
U 1AA0	9436:	9437:	9438:	9440:	9441:	9442:	9443:	9444:
U 1AA8	9445:	9446:	9447:	9448:	9449:	9450:	9451:	9453:
U 1AB0	9454:	9456:	9457:	9458:	9459:	9460:	9461:	9462:
U 1AB8	9463:	9464:	9465:	9466:	9467:	9468:	9469:	9470:
U 1AC0	9471:	9472:	9473:	9474:	9476:	9477:	9479:	9480:
U 1AC8	9481:	9482:	9483:	9484:	9485:	9486:	9487:	9488:
U 1AD0	9489:	9490:	9491:	9492:	9493:	9494:	9495:	9496:
U 1AD8	9497:	9499:	9500:	9501:	9502:	9503:	9504:	9505:
U 1AE0	9507:	9508:	9509:	9510:	9511:	9512:	9513:	9515:
U 1AE8	9516:	9517:	9518:	9519:	9520:	9521:	9523:	9524:
U 1AF0	9525:	9526:	9527:	9528:	9529:	9531:	9532:	9533:
U 1AF8	9535:	9536:	9537:	9538:	9539:	9540:	9541:	9542:
U 1B00	9543:	9544:	9545:	9546:	9547:	9548:	9549:	9550:
U 1B08	9552:	9553:	9554:	9556:	9557:	9558:	9559:	9560:
U 1B10	9561:	9562:	9563:	9564:	9565:	9566:	9567:	9568:
U 1B18	9569:	9570:	9571:	9572:	9573:	9574:	9575:	9576:

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 1CD0	10044:	10045:	10046:	10047:	10049:	10050:	10051:	10052:
U 1CD8	10053:	10054:	10055:	10056:	10057:	10059:	10060:	10061:
U 1CE0	10062:	10063:	10064:	10065:	10066:	10067:	10069:	10070:
U 1CE8	10071:	10072:	10073:	10074:	10075:	10076:	10077:	10079:
U 1CF0	10080:	10081:	10082:	10083:	10084:	10085:	10086:	10087:
U 1CF8	10089:	10090:	10091:	10092:	10093:	10094:	10095:	10096:
U 1D00	10097:	10099:	10100:	10101:	10102:	10103:	10104:	10105:
U 1D08	10106:	10107:	10109:	10110:	10111:	10112:	10113:	10114:
U 1D10	10115:	10116:	10117:	10119:	10120:	10121:	10122:	10123:
U 1D18	10124:	10125:	10126:	10127:	10129:	10130:	10131:	10132:
U 1D20	10133:	10134:	10135:	10136:	10137:	10139:	10140:	10141:
U 1D28	10142:	10143:	10144:	10145:	10146:	10147:	10148:	10149:
U 1D30	10151:	10152:	10153:	10154:	10155:	10156:	10157:	10158:
U 1D38	10159:	10160:	10161:	10163:	10164:	10165:	10166:	10167:
U 1D40	10168:	10169:	10170:	10171:	10172:	10173:	10175:	10176:
U 1D48	10177:	10178:	10179:	10180:	10181:	10182:	10183:	10184:
U 1D50	10185:	10187:	10188:	10189:	10190:	10191:	10192:	10193:
U 1D58	10194:	10195:	10196:	10198:	10199:	10200:	10201:	10202:
U 1D60	10203:	10204:	10205:	10206:	10207:	10209:	10210:	10211:
U 1D68	10212:	10213:	10214:	10215:	10216:	10217:	10218:	10220:
U 1D70	10221:	10222:	10223:	10224:	10225:	10226:	10227:	10228:
U 1D78	10229:	10231:	10232:	10233:	10234:	10235:	10236:	10237:
U 1D80	10238:	10239:	10240:	10242:	10243:	10244:	10245:	10246:
U 1D88	10247:	10248:	10249:	10250:	10251:	10253:	10254:	10255:
U 1D90	10256:	10257:	10258:	10259:	10260:	10261:	10262:	10264:
U 1D98	10265:	10266:	10267:	10268:	10269:	10270:	10271:	10272:
U 1DA0	10273:	10275:	10276:	10277:	10278:	10279:	10280:	10281:
U 1DA8	10282:	10283:	10284:	10286:	10287:	10288:	10289:	10290:
U 1DB0	10291:	10292:	10293:	10294:	10295:	10297:	10298:	10299:
U 1DB8	10300:	10301:	10302:	10303:	10304:	10305:	10306:	10308:
U 1DC0	10309:	10310:	10311:	10312:	10313:	10314:	10315:	10316:
U 1DC8	10317:	10319:	10320:	10321:	10322:	10323:	10324:	10325:
U 1DD0	10326:	10327:	10328:	10329:	10331:	10332:	10333:	10334:
U 1DD8	10335:	10336:	10337:	10338:	10339:	10340:	10341:	10343:
U 1DE0	10344:	10345:	10346:	10347:	10348:	10349:	10350:	10351:
U 1DE8	10352:	10353:	10355:	10356:	10357:	10358:	10359:	10360:
U 1DF0	10361:	10362:	10363:	10364:	10365:	10367:	10368:	10369:
U 1DF8	10370:	10371:	10372:	10373:	10374:	10375:	10376:	10377:
U 1E00	10379:	10380:	10381:	10382:	10383:	10384:	10385:	10386:
U 1E08	10387:	10388:	10389:	10391:	10392:	10393:	10394:	10395:
U 1E10	10396:	10397:	10398:	10399:	10400:	10401:	10403:	10404:
U 1E18	10405:	10406:	10407:	10408:	10409:	10410:	10411:	10412:
U 1E20	10413:	10415:	10416:	10417:	10418:	10419:	10420:	10421:
U 1E28	10422:	10423:	10424:	10426:	10427:	10428:	10429:	10430:
U 1E30	10431:	10432:	10433:	10434:	10435:	10437:	10438:	10439:
U 1E38	10440:	10441:	10442:	10443:	10444:	10445:	10446:	10448:
U 1E40	10449:	10450:	10451:	10452:	10453:	10454:	10455:	10456:
U 1E48	10457:	10458:	10459:	10461:	10462:	10463:	10464:	10466:
U 1E50	10467:	10468:	10469:	10470:	10471:	10473:	10474:	10475:
U 1E58	10476:	10478:	10479:	10480:	10481:	10482:	10483:	10485:
U 1E60	10486:	10487:	10488:	10490:	10491:	10492:	10493:	10494:
U 1E68	10495:	10497:	10498:	10499:	10500:	10502:	10503:	10504:
U 1E70	10505:	10506:	10507:	10509:	10510:	10511:	10512:	10514:

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 1E78	10515:	10516:	10517:	10518:	10519:	10521:	10522:	10523:
U 1E80	10524:	10526:	10527:	10528:	10529:	10530:	10531:	10533:
U 1E88	10534:	10535:	10536:	10538:	10539:	10540:	10541:	10542:
U 1E90	10543:	10545:	10546:	10547:	10548:	10550:	10551:	10552:
U 1E98	10553:	10554:	10555:	10556:	10558:	10559:	10560:	10561:
U 1EA0	10562:	10563:	10564:	10565:	10566:	10568:	10569:	10570:
U 1EA8	10571:	10572:	10573:	10574:	10575:	10576:	10578:	10579:
U 1EB0	10580:	10581:	10582:	10583:	10584:	10585:	10586:	10588:
U 1EB8	10589:	10590:	10591:	10592:	10593:	10594:	10595:	10596:
U 1EC0	10598:	10599:	10600:	10601:	10602:	10603:	10604:	10605:
U 1EC8	10606:	10608:	10609:	10610:	10611:	10612:	10613:	10614:
U 1ED0	10615:	10616:	10618:	10619:	10620:	10621:	10622:	10623:
U 1ED8	10624:	10625:	10626:	10628:	10629:	10630:	10631:	10632:
U 1EE0	10633:	10634:	10635:	10636:	10638:	10639:	10640:	10641:
U 1EE8	10642:	10643:	10644:	10645:	10646:	10648:	10649:	10650:
U 1EF0	10651:	10652:	10653:	10654:	10655:	10656:	10658:	10659:
U 1EF8	10660:	10661:	10662:	10663:	10664:	10665:	10666:	10668:
U 1F00	10669:	10670:	10671:	10672:	10673:	10674:	10675:	10676:
U 1F08	10678:	10679:	10680:	10681:	10682:	10683:	10684:	10685:
U 1F10	10686:	10687:	10689:	10690:	10691:	10692:	10693:	10694:
U 1F18	10695:	10696:	10697:	10698:	10700:	10701:	10702:	10703:
U 1F20	10704:	10705:	10706:	10707:	10708:	10709:	10711:	10712:
U 1F28	10713:	10714:	10715:	10716:	10717:	10718:	10719:	10720:
U 1F30	10722:	10723:	10724:	10725:	10726:	10727:	10728:	10729:
U 1F38	10730:	10732:	10733:	10734:	10735:	10736:	10737:	10738:
U 1F40	10739:	10740:	10742:	10743:	10744:	10745:	10746:	10747:
U 1F48	10748:	10749:	10750:	10752:	10753:	10754:	10755:	10756:
U 1F50	10757:	10758:	10759:	10760:	10762:	10763:	10764:	10765:
U 1F58	10766:	10767:	10768:	10769:	10770:	10772:	10773:	10774:
U 1F60	10775:	10776:	10777:	10778:	10779:	10780:	10782:	10783:
U 1F68	10784:	10785:	10786:	10787:	10788:	10789:	10790:	10792:
U 1F70	10793:	10794:	10795:	10796:	10797:	10798:	10799:	10800:
U 1F78	10802:	10803:	10804:	10805:	10806:	10807:	10808:	10809:
U 1F80	10810:	10811:	10813:	10814:	10815:	10816:	10817:	10818:
U 1F88	10819:	10820:	10821:	10822:	10823:			
U 1F90 - 1FF7 Unused								
U 1FF8							6216:	6217:
U 2000	6218:		10826:	10827:	10828:	10829:	10830:	10831:
U 2008	10832:	10833:	10834:	10835:	10836:			
U 2010 - 2017 Unused								
U 2018	6336:		10839:	10840:	10841:	10842:	10843:	10844:
U 2020	10845:	10846:	10847:	10848:	10849:	10850:	10852:	10853:
U 2028	10854:	10856:	10857:	10858:	10859:	10860:	10861:	10863:
U 2030	10864:	10865:	10867:	10868:	10869:	10870:	10871:	10872:
U 2038	10874:	10875:	10876:	10878:	10879:	10880:	10881:	10882:
U 2040	10883:	10885:	10886:	10887:	10889:	10890:	10891:	10892:
U 2048	10893:	10894:	10895:	10897:	10898:	10899:	10900:	10901:
U 2050	10902:	10903:	10904:	10905:	10907:	10908:	10909:	10910:
U 2058	10911:	10912:	10913:	10914:	10915:	10917:	10918:	10919:
U 2060	10920:	10921:	10922:	10923:	10924:	10925:	10927:	10928:
U 2068	10929:	10930:	10931:	10932:	10933:	10934:	10935:	10936:
U 2070	10937:	10938:	10940:	10941:	10942:	10944:	10945:	10946:
U 2078	10947:	10948:	10949:	10951:	10952:	10953:	10955:	10956:

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 2080	10957:	10958:	10959:	10960:	10962:	10963:	10964:	10966:
U 2088	10967:	10968:	10969:	10970:	10971:	10973:	10974:	10975:
U 2090	10977:	10978:	10979:	10980:	10981:	10982:	10984:	10985:
U 2098	10986:	10988:	10989:	10990:	10991:	10992:	10993:	10995:
U 20A0	10996:	10997:	10999:	11000:	11001:	11002:	11003:	11004:
U 20A8	11006:	11007:	11008:	11010:	11011:	11012:	11013:	11014:
U 20B0	11015:	11017:	11018:	11019:	11021:	11022:	11023:	11024:
U 20B8	11025:	11026:	11028:	11029:	11030:	11032:	11033:	11034:
U 20C0	11035:	11036:	11037:	11039:	11040:	11041:	11043:	11044:
U 20C8	11045:	11046:	11047:	11048:	11050:	11051:	11052:	11054:
U 20D0	11055:	11056:	11057:	11058:	11059:	11061:	11062:	11063:
U 20D8	11065:	11066:	11067:	11068:	11069:	11070:	11071:	11073:
U 20E0	11074:	11075:	11076:	11077:	11078:	11079:	11080:	11082:
U 20E8	11083:	11084:	11085:	11086:	11087:	11088:	11089:	11091:
U 20F0	11092:	11093:	11094:	11095:	11096:	11097:	11098:	11100:
U 20F8	11101:	11102:	11103:	11104:	11105:	11106:	11107:	11109:
U 2100	11110:	11111:	11112:	11113:	11114:	11115:	11116:	11118:
U 2108	11119:	11120:	11121:	11122:	11123:	11124:	11125:	11127:
U 2110	11128:	11129:	11130:	11131:	11132:	11133:	11134:	11136:
U 2118	11137:	11138:	11139:	11140:	11141:	11142:	11143:	11145:
U 2120	11146:	11147:	11148:	11149:	11150:	11151:	11153:	11154:
U 2128	11155:	11156:	11157:	11158:	11159:	11161:	11162:	11163:
U 2130	11164:	11165:	11166:	11167:	11169:	11170:	11171:	11172:
U 2138	11173:	11174:	11175:	11177:	11178:	11179:	11180:	11181:
U 2140	11182:	11183:	11185:	11186:	11187:	11188:	11189:	11190:
U 2148	11191:	11193:	11194:	11195:	11196:	11197:	11198:	11199:
U 2150	11201:	11202:	11203:	11204:	11205:	11206:	11207:	11210:
U 2158	11211:	11212:	11214:	11215:	11216:	11217:	11218:	11219:
U 2160	11220:	11221:	11222:	11224:	11225:	11226:	11228:	11229:
U 2168	11230:	11231:	11232:	11233:	11234:	11235:	11236:	11238:
U 2170	11239:	11240:	11242:	11243:	11244:	11245:	11246:	11247:
U 2178	11248:	11249:	11250:	11252:	11253:	11254:	11256:	11257:
U 2180	11258:	11259:	11260:	11261:	11262:	11263:	11264:	11266:
U 2188	11267:	11268:	11270:	11271:	11272:	11273:	11274:	11275:
U 2190	11276:	11277:	11278:	11280:	11281:	11282:	11284:	11285:
U 2198	11286:	11287:	11288:	11289:	11290:	11291:	11292:	11294:
U 21A0	11295:	11296:	11298:	11299:	11300:	11301:	11302:	11303:
U 21A8	11304:	11305:	11306:	11308:	11309:	11310:	11312:	11313:
U 21B0	11314:	11315:	11316:	11317:	11318:	11319:	11320:	11322:
U 21B8	11323:	11324:	11326:	11327:	11328:	11329:	11330:	11331:
U 21C0	11332:	11333:	11334:	11335:	11337:	11338:	11339:	11341:
U 21C8	11342:	11343:	11344:	11345:	11346:	11347:	11348:	11349:
U 21D0	11350:	11352:	11353:	11354:	11356:	11357:	11358:	11359:
U 21D8	11360:	11361:	11362:	11363:	11364:	11365:	11367:	11368:
U 21E0	11369:	11371:	11372:	11373:	11374:	11375:	11376:	11377:
U 21E8	11378:	11379:	11380:	11382:	11383:	11384:	11386:	11387:
U 21F0	11388:	11389:	11390:	11391:	11392:	11393:	11394:	11396:
U 21F8	11397:	11398:	11400:	11401:	11402:	11403:	11404:	11405:
U 2200	11406:	11407:	11408:	11410:	11411:	11412:	11414:	11415:
U 2208	11416:	11417:	11418:	11419:	11420:	11421:	11422:	11424:
U 2210	11425:	11426:	11428:	11429:	11430:	11431:	11432:	11433:
U 2218	11434:	11435:	11436:	11438:	11439:	11440:	11441:	11442:
U 2220	11443:	11444:	11445:	11446:	11447:	11448:	11449:	11451:

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 2228	11452:	11453:	11454:	11455:	11456:	11457:	11458:	11459:
U 2230	11460:	11461:	11462:	11464:	11465:	11466:	11467:	11468:
U 2238	11469:	11470:	11471:	11472:	11473:	11474:	11475:	11477:
U 2240	11478:	11479:	11480:	11481:	11482:	11483:	11484:	11485:
U 2248	11486:	11487:	11488:	11490:	11491:	11492:	11493:	11494:
U 2250	11495:	11496:	11497:	11499:	11500:	11501:	11502:	11503:
U 2258	11504:	11505:	11506:	11508:	11509:	11510:	11511:	11512:
U 2260	11513:	11514:	11515:	11517:	11518:	11519:	11520:	11521:
U 2268	11522:	11523:	11524:	11526:	11527:	11528:	11529:	11530:
U 2270	11531:	11532:	11533:	11534:	11535:	11536:	11538:	11539:
U 2278	11540:	11541:	11542:	11543:	11544:	11545:	11546:	11547:
U 2280	11548:	11550:	11551:	11552:	11553:	11554:	11555:	11556:
U 2288	11557:	11558:	11559:	11560:	11562:	11563:	11564:	11565:
U 2290	11566:	11567:	11568:	11569:	11570:	11571:	11572:	11574:
U 2298	11575:	11576:	11577:	11578:	11579:	11580:	11581:	11582:
U 22A0	11583:	11584:	11586:	11587:	11588:	11589:	11590:	11591:
U 22A8	11592:	11593:	11594:	11595:	11596:	11598:	11599:	11600:
U 22B0	11601:	11602:	11603:	11604:	11605:	11606:	11607:	11608:
U 22B8	11610:	11611:	11612:	11613:	11614:	11615:	11616:	11617:
U 22C0	11618:	11619:	11620:	11622:	11623:	11624:	11625:	11626:
U 22C8	11627:	11628:	11629:	11630:	11632:	11633:	11634:	11635:
U 22D0	11636:	11637:	11638:	11639:	11640:	11642:	11643:	11644:
U 22D8	11645:	11646:	11647:	11648:	11649:	11650:	11652:	11653:
U 22E0	11654:	11655:	11656:	11657:	11658:	11659:	11660:	11662:
U 22E8	11663:	11664:	11665:	11666:	11667:	11668:	11669:	11670:
U 22F0	11672:	11673:	11674:	11675:	11676:	11677:	11678:	11679:
U 22F8	11680:	11682:	11683:	11684:	11685:	11686:	11687:	11688:
U 2300	11689:	11690:	11692:	11693:	11694:	11695:	11696:	11697:
U 2308	11698:	11699:	11700:	11702:	11703:	11704:	11705:	11706:
U 2310	11707:	11708:	11709:	11710:	11712:	11713:	11714:	11715:
U 2318	11716:	11717:	11718:	11719:	11720:	11722:	11723:	11724:
U 2320	11725:	11726:	11727:	11728:	11729:	11730:	11732:	11733:
U 2328	11734:	11735:	11736:	11737:	11738:	11739:	11740:	11742:
U 2330	11743:	11744:	11745:	11746:	11747:	11748:	11749:	11750:
U 2338	11752:	11753:	11754:	11755:	11756:	11757:	11758:	11759:
U 2340	11760:	11762:	11763:	11764:	11765:	11766:	11767:	11768:
U 2348	11769:	11770:	11772:	11773:	11774:	11775:	11776:	11777:
U 2350	11778:	11779:	11780:	11782:	11783:	11784:	11785:	11786:
U 2358	11787:	11788:	11789:	11790:	11792:	11793:	11794:	11795:
U 2360	11796:	11797:	11798:	11799:	11800:	11802:	11803:	11804:
U 2368	11805:	11806:	11807:	11808:	11809:	11810:	11812:	11813:
U 2370	11814:	11815:	11816:	11817:	11818:	11819:	11820:	11822:
U 2378	11823:	11824:	11825:	11826:	11827:	11828:	11829:	11830:
U 2380	11831:	11833:	11834:	11835:	11836:	11837:	11838:	11839:
U 2388	11840:	11841:	11842:	11844:	11845:	11846:	11847:	11848:
U 2390	11849:	11850:	11851:	11852:	11853:	11855:	11856:	11857:
U 2398	11858:	11859:	11860:	11861:	11862:	11863:	11864:	11866:
U 23A0	11867:	11868:	11869:	11870:	11871:	11872:	11873:	11874:
U 23A8	11875:	11877:	11878:	11879:	11880:	11881:	11882:	11883:
U 23B0	11884:	11885:	11886:	11888:	11889:	11890:	11891:	11892:
U 23B8	11893:	11894:	11895:	11896:	11897:	11899:	11900:	11901:
U 23C0	11902:	11903:	11904:	11905:	11906:	11907:	11908:	11910:
U 23C8	11911:	11912:	11913:	11914:	11915:	11916:	11917:	11918:

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 23D0	11920:	11921:	11922:	11923:	11924:	11925:	11926:	11927:
U 23D8	11928:	11930:	11931:	11932:	11933:	11934:	11935:	11936:
U 23E0	11937:	11938:	11940:	11941:	11942:	11943:	11944:	11945:
U 23E8	11946:	11947:	11948:	11950:	11951:	11952:	11953:	11954:
U 23F0	11955:	11956:	11957:	11958:	11960:	11961:	11962:	11963:
U 23F8	11964:	11965:	11966:	11967:	11968:	11970:	11971:	11972:
U 2400	11973:	11974:	11975:	11976:	11977:	11978:	11980:	11981:
U 2408	11982:	11983:	11984:	11985:	11986:	11987:	11988:	11989:
U 2410 - 2417	Unused							
U 2418	6339:		11992:	11993:	11994:	11995:	11996:	11997:
U 2420	11998:	11999:	12000:	12002:	12003:	12004:	12005:	12006:
U 2428	12007:	12008:	12009:	12010:	12012:	12013:	12014:	12015:
U 2430	12016:	12017:	12018:	12019:	12020:	12022:	12023:	12024:
U 2438	12025:	12026:	12027:	12028:	12029:	12030:	12032:	12033:
U 2440	12034:	12035:	12036:	12037:	12038:	12039:	12040:	12042:
U 2448	12043:	12044:	12045:	12046:	12047:	12048:	12049:	12050:
U 2450	12051:	12053:	12054:	12055:	12056:	12057:	12058:	12059:
U 2458	12060:	12061:	12063:	12064:	12065:	12066:	12067:	12068:
U 2460	12069:	12070:	12071:	12073:	12074:	12075:	12076:	12077:
U 2468	12078:	12079:	12080:	12081:	12082:	12084:	12085:	12086:
U 2470	12087:	12088:	12089:	12090:	12091:	12092:	12093:	12095:
U 2478	12096:	12097:	12098:	12099:	12100:	12101:	12102:	12103:
U 2480	12104:	12106:	12107:	12108:	12109:	12110:	12111:	12112:
U 2488	12113:	12114:	12115:	12117:	12118:	12119:	12120:	12121:
U 2490	12122:	12123:	12124:	12125:	12126:	12128:	12129:	12130:
U 2498	12131:	12132:	12133:	12134:	12135:	12136:	12137:	12139:
U 24A0	12140:	12141:	12142:	12143:	12144:	12145:	12146:	12147:
U 24A8	12148:	12150:	12151:	12152:	12153:	12154:	12155:	12156:
U 24B0	12157:	12158:	12159:	12161:	12162:	12163:	12164:	12165:
U 24B8	12166:	12167:	12168:	12169:	12171:	12172:	12173:	12174:
U 24C0	12175:	12176:	12177:	12178:	12179:	12181:	12182:	12183:
U 24C8	12184:	12185:	12186:	12187:	12188:	12189:	12191:	12192:
U 24D0	12193:	12194:	12195:	12196:	12197:	12198:	12199:	12201:
U 24D8	12202:	12203:	12204:	12205:	12206:	12207:	12209:	12210:
U 24E0	12211:	12212:	12213:	12214:	12215:	12217:	12218:	12219:
U 24E8	12220:	12221:	12222:	12223:	12225:	12226:	12227:	12228:
U 24F0	12229:	12230:	12231:	12233:	12234:	12235:	12236:	12237:
U 24F8	12238:	12239:	12241:	12242:	12243:	12244:	12245:	12246:
U 2500	12247:	12249:	12250:	12251:	12252:	12253:	12254:	12255:
U 2508	12257:	12258:	12259:	12260:	12261:	12262:	12263:	12265:
U 2510	12266:	12267:	12268:	12269:	12270:	12271:	12273:	12274:
U 2518	12275:	12276:	12277:	12278:	12279:	12281:	12282:	12283:
U 2520	12284:	12285:	12286:	12287:	12289:	12290:	12291:	12292:
U 2528	12293:	12294:	12295:	12297:	12298:	12299:	12300:	12301:
U 2530	12302:	12303:	12305:	12306:	12307:	12308:	12309:	12310:
U 2538	12311:	12313:	12314:	12315:	12316:	12317:	12318:	12319:
U 2540	12321:	12322:	12323:	12324:	12325:	12326:	12327:	12329:
U 2548	12330:	12331:	12332:	12333:	12334:	12335:	12336:	12338:
U 2550	12339:	12340:	12341:	12342:	12343:	12344:	12345:	12347:
U 2558	12348:	12349:	12350:	12351:	12352:	12353:	12354:	12356:
U 2560	12357:	12358:	12359:	12360:	12361:	12362:	12363:	12365:
U 2568	12366:	12367:	12368:	12369:	12370:	12371:	12372:	12374:
U 2570	12375:	12376:	12377:	12378:	12379:	12380:	12381:	12383:

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 2578	12384:	12385:	12386:	12387:	12388:	12389:	12390:	12392:
U 2580	12393:	12394:	12395:	12396:	12397:	12398:	12399:	12401:
U 2588	12402:	12403:	12404:	12405:	12406:	12407:	12408:	12410:
U 2590	12411:	12412:	12413:	12414:	12415:	12416:	12417:	12419:
U 2598	12420:	12421:	12422:	12423:	12424:	12425:	12426:	12428:
U 25A0	12429:	12430:	12431:	12432:	12433:	12434:	12435:	12437:
U 25A8	12438:	12439:	12440:	12441:	12442:	12443:	12444:	12446:
U 25B0	12447:	12448:	12449:	12450:	12451:	12452:	12453:	12455:
U 25B8	12456:	12457:	12458:	12459:	12460:	12461:	12462:	12464:
U 25C0	12465:	12466:	12467:	12468:	12469:	12470:	12471:	12473:
U 25C8	12474:	12475:	12476:	12477:	12478:	12479:	12480:	12482:
U 25D0	12483:	12484:	12485:	12486:	12487:	12488:	12489:	12491:
U 25D8	12492:	12493:	12494:	12495:	12496:	12497:	12498:	12500:
U 25E0	12501:	12502:	12503:	12504:	12505:	12506:	12507:	12509:
U 25E8	12510:	12511:	12512:	12513:	12514:	12515:	12516:	12518:
U 25F0	12519:	12520:	12521:	12522:	12523:	12524:	12525:	12527:
U 25F8	12528:	12529:	12530:	12531:	12532:	12533:	12534:	12536:
U 2600	12537:	12538:	12539:	12540:	12541:	12542:	12543:	12545:
U 2608	12546:	12547:	12548:	12549:	12550:	12551:	12552:	12554:
U 2610	12555:	12556:	12557:	12558:	12559:	12560:	12561:	12563:
U 2618	12564:	12565:	12566:	12567:	12568:	12569:	12570:	12572:
U 2620	12573:	12574:	12575:	12576:	12577:	12578:	12579:	12581:
U 2628	12582:	12583:	12584:	12585:	12586:	12587:	12588:	12590:
U 2630	12591:	12592:	12593:	12594:	12595:	12596:	12597:	12599:
U 2638	12600:	12601:	12602:	12603:	12604:	12605:	12606:	12608:
U 2640	12609:	12610:	12611:	12612:	12613:	12614:	12615:	12617:
U 2648	12618:	12619:	12620:	12621:	12622:	12623:	12624:	12626:
U 2650	12627:	12628:	12629:	12630:	12631:	12632:	12633:	12635:
U 2658	12636:	12637:	12638:	12639:	12640:	12641:	12642:	12644:
U 2660	12645:	12646:	12647:	12648:	12649:	12650:	12651:	12653:
U 2668	12654:	12655:	12656:	12657:	12658:	12659:	12660:	12662:
U 2670	12663:	12664:	12665:	12666:	12667:	12668:	12669:	12671:
U 2678	12672:	12673:	12674:	12675:	12676:	12677:	12678:	12680:
U 2680	12681:	12682:	12683:	12684:	12685:	12686:	12687:	12689:
U 2688	12690:	12691:	12692:	12693:	12694:	12695:	12696:	12698:
U 2690	12699:	12700:	12701:	12702:	12703:	12704:	12705:	12707:
U 2698	12708:	12709:	12710:	12711:	12712:	12713:	12714:	12716:
U 26A0	12717:	12718:	12719:	12720:	12721:	12722:	12723:	12725:
U 26A8	12726:	12727:	12728:	12729:	12730:	12731:	12732:	12734:
U 26B0	12735:	12736:	12737:	12738:	12739:	12740:	12741:	12743:
U 26B8	12744:	12745:	12746:	12747:	12748:	12749:	12750:	12752:
U 26C0	12753:	12754:	12755:	12756:	12757:	12758:	12759:	12761:
U 26C8	12762:	12763:	12764:	12765:	12766:	12767:	12768:	12770:
U 26D0	12771:	12772:	12773:	12774:	12775:	12776:	12777:	12779:
U 26D8	12780:	12781:	12782:	12783:	12784:	12785:	12786:	12788:
U 26E0	12789:	12790:	12791:	12792:	12793:	12794:	12795:	12797:
U 26E8	12798:	12799:	12800:	12801:	12802:	12803:	12804:	12806:
U 26F0	12807:	12808:	12809:	12810:	12811:	12812:	12813:	12815:
U 26F8	12816:	12817:	12818:	12819:	12820:	12821:	12822:	12824:
U 2700	12825:	12826:	12827:	12828:	12829:	12830:	12831:	12833:
U 2708	12834:	12835:	12836:	12837:	12838:	12839:	12840:	12842:
U 2710	12843:	12844:	12845:	12846:	12847:	12848:	12849:	12851:
U 2818	12852:	12853:	12854:	12855:	12856:	12857:	12858:	12860:

- 2817 Unused
 6342:

ZZ-ENKCB-1.0 ;
; ENKCB.MCR
;

MICRO2 1L(03) Location / Line Num 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Rev 01.00
Location / Line Number Index

1 7
Fiche 2 Frame 17
Sequence 292
Page 286

	Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U	2820 - 2A7F	Unused							
U	2A80	6515:	6516:	6517:	6518:	6520:	6521:		
U	2A88 - 2C17	Unused							
U	2C18	6345:							
U	2C20 - 2FF7	Unused							
U	2FF8							6222:	6223:
U	3000	6224:							
U	3008 - 3017	Unused							
U	3018	6348:							
U	3020 - 3417	Unused							
U	3418	6351:							
U	3420 - 35F7	Unused							
U	35F8					6278:	6279:	6280:	
U	3600 - 36F7	Unused							
U	36F8						6283:	6284:	
U	3700 - 37F7	Unused							
U	37F8							6228:	6229:
U	3800	6230:							
U	3808 - 3817	Unused							
U	3818	6354:							
U	3820 - 3BF7	Unused							
U	3BF8							6233:	6234:
U	3C00	6235:							
U	3C08 - 3C17	Unused							
U	3C18	6357:							
U	3C20 - 3DF7	Unused							
U	3DF8							6238:	6239:
U	3E00	6240:							
U	3E08 - 3EF7	Unused							
U	3EF8							6243:	6244:
U	3F00	6245:							
U	3F08 - 3F77	Unused							
U	3F78							6248:	6249:
U	3F80	6250:							
U	3F88 - 3FB7	Unused							
U	3FB8							6253:	6254:
U	3FC0	6255:							
U	3FC8 - 3FD7	Unused							
U	3FD8							6258:	6259:
U	3FE0	6260:							
U	3FE8							6263:	6264:
U	3FF0	6265:						6268:	6269:
U	3FF8	6270:	6273:	6274:	6275:				6286:

ZZ-ENKCB-1.0 ;
: ENKCB.MCR
:
MICRO2 1L(03) C-code Microword Su 3-MAR-82 J 7
C-code Microword Summary 3-MAR-82 10:02:46 ENKCB Micro-diagnostic for DAP module Rev 01.00 Sequence 293 Page 287

ENKCB Words not
in bounds
512

Used 512
Remaining

Total microwords used in memory C: 512
Total microwords remaining in memory C: 0
Highest address used in memory C: 180 (hex)

ENKCB Words not
in bounds 5350

Used 5350
Remaining

Total microwords used in memory U: 5350
Total microwords remaining in memory U: 0
Highest address used in memory U: 3FFF (hex)

ZZ-ENKCB-1.0 :
: ENKCB.MCR
:
MICRO2 1L(03)
Error Summary

Error Summary 3-MAR-82 10:02:46 L 7
3-MAR-82 ENKCB Micro-diagnostic for DAP module

Fiche 2 Frame L7
Sequence 295
Rev 01.00 Page 289

6216 Validity failure in file ENKCB.MIC
JSR [SUB.T1] ;PUSH RET ADD OF 1FFF ONTO STAC
6222 Validity failure in file ENKCB.MIC
JSR [SUB.T1] ;PUSH RET ADD OF 2FFF ONTO STAC
6228 Validity failure in file ENKCB.MIC
JSR [SUB.T1] ;PUSH RET ADD OF 37FF ONTO STAC

Pass 1 warnings: 0 Pass 2 warnings: 3
Pass 1 errors: 0 Pass 2 errors: 0